



# **User Manual**

**Version 5.2.6**

**November 11, 2004**

**Sand Hill Engineering  
Box 517 Geneva, Florida 32732**

Expect amazing things !



## Copyright and Disclaimer

The application XTension, its design concepts and product design and this manual, and logo are Copyright ©2004 Sand Hill Engineering. Product names mentioned herein are copyright or trademarks of their respective owners.

XTension is a program specifically designed for small to moderate sized automation and security applications.

XTension is in itself a development system, and as such it is possible for you to create custom systems which may be syntactically correct, but are illogical or incomplete.

XTension is only a part of a system which potentially includes hundreds of electrical devices. The probability of failure of any part of the system is real.

For this reason, Sand Hill Engineering cannot be held responsible for any loss you might incur due to failure of operation of any part of your automation system.



## Why we think you will like XTension

XTension is a control and monitor application for all types of process control, from home automation to the mid-scale factory. XTension provides a single, familiar and encouraging system where you can integrate all of your schedules, procedures, data collection and device control.

Much of the experience gained in the development and operation of NASA's Space Shuttle Launch Processing System is incorporated into XTension. The LPS has been in daily use since 1978, serving more than 120,000 control and monitor points.

XTension uses the best of the OS X standards to bring you a well organized and thoughtfully presented home automation agent. You can be sure that XTension will continue to honor these standards and attempt to follow new MacOS features as they evolve.

Rather than create another proprietary scripting language, XTension uses the MacOS standard AppleScript, which has an established grammar. Assertions of commitment by Apple promise that AppleScript will continue to grow with the MacOS. This means that XTension will also continue to grow without the need for continuous updates.

You may notice that XTension is both fast and efficient. It requires very little hard drive space and can in a memory partition of as little as 500KB. We think you will find this pleasing in this day of multi-megabyte applications.

Another issue is that of longevity. In that XTension will run on any Macintosh\* known as of May 2004, it is important that XTension remain compatible with a wide variety of 'closet Macs'.

There are many Macs still running well after 15 or more years, and it is likely that the ones we have today will last at least that long.

Although many XTension users enjoy running on a fast Mac with other applications, it is most common to see the whole home automation process relegated to an old MacSE/30 or so and just let it be.

Ten years from now, our G5's will probably be the 'closet Macs' and you'll still be able to run XTension on them...

\* Of course the Mac 128 and 512 won't run OS 7.1...

# Getting started

---

## If you don't read manuals:

- Do read the “What you need” section.
- Do look at some example scripts, and notice the dictionary.  
(Use the Script Editor to view it if you don't want to read it here)
- Browse some of the menus, buttons and scripts of the Demo database.
- Read the next chapter on Just the Facts. If you understand it, then you are on your way. Otherwise, study the manual, and the tutorials and articles on the website ([www.shed.com](http://www.shed.com)).
- PLEASE Note that XTension must be active at all times.  
If you shut down your Mac, XTension cannot function.
- XTension does **not** download to the CM11A.
- Please do not use the Demo Database as the foundation of your system. Always start with a 'fresh' database :

On startup, if XTension cannot find a file named XTension Database, it will create a new empty one... so just rename the Demo database, or move it out of the XTension folder.  
the latest help for all sorts of problems is at :  
[www.shed.com/ddt.html](http://www.shed.com/ddt.html)

## What you need

- **Macintosh** running System 10.2 or later.
- **One of the supported X-10 interfaces :**  
This is a small box which makes it easier for the host Macintosh to send and receive X-10 messages to and from the power bus:
  - **'ActiveHome'** interface (CM11A) from X-10 (USA).  
All international versions of the CM11 are also supported.  
XTension does not support download to the CM11A.  
(The CM11a is sold under various brand names, but all look alike)
  - **'Any version of the LynX'** interface from Marrick Ltd.  
This includes the LynX-10, LynX-Port, and the LynX-PLC.  
(XTension does not support all functions of the 'LynX-Port'.)
- **Interface cable ( Serial or USB )**

When you purchase one of the X10 interfaces, you should also purchase the cable that is offered. Normally these devices are 'serial' and assume that you are connecting to a "PC". This is ok since you are going to be either connecting to a USB, or a serial adaptor of some sort on the Mac.

## • X-10 devices

Any device which sends or receives X-10 commands, will work with XTension.

However, do be aware that X-10 does make a series of "Security Consoles"



which use sensors and controls that are not compatible with their Home Automation devices.

X-10 and their dealers do try to distinguish the difference in their advertising and catalogs.

## **Making the connection**

- **If you have a LynX-10 or LynX-Port interface :**

- Plug in TW-523 to a wall socket near the 'LynX'
- Connect the TW-523 cable to the interface.
- Connect the serial cable between the Macintosh serial port or USB adaptor and the interface.

- **If you have a CM11A or LynX-PLC :**

- Plug in the interface to a wall socket near your Macintosh.
- Plug in the serial cable between the interface or USB adaptor and the Mac.  
( DO NOT put the batteries in the CM11, you don't need them )



## Install the program

---

The **installation package** for XTension contains all items necessary, and needs only to be copied to your hard disk.

**PLEASE** resist the temptation to put the application directly into your Applications folder. Although you may certainly put the entire XTension folder into your applications folder, we suggest that you consider that XTension is a distinctively different kind of program, and should be allowed to occupy a folder of its own directly at the top level of your disk.

Note that license to XTension allows multiple copies in a single household for a single interface.

XTension will create other files and folders in the XTension folder when you run the program for the first time :

- XTension Database
- XTension Log
- Views folder : graphic images for View backgrounds
- Icons folder : graphic images for Unit Icons
- Logs folder : saved Log Files

Please note that your Preferences are kept in the XTension Database.



## I think I Got It, Just get me going :

Or, if the manual is just too much,  
how do I just make things GO ? :

**If this doesn't make sense, just keep going through the manual....**

**AND Please see the tutorials and articles at [WWW.SHED.COM](http://WWW.SHED.COM).**

- You should think about what you want to do with your whole house, even if you don't intend to install everything at once.

This will save a lot of time and bother later. Be most careful if you are a programmer, it is dangerous to substitute skill with the coding pencil for just a little forethought...

### **Just the simplest elements :**

- Once you have started up XTension, choose New Unit from the File Menu...
- Give a name to the new unit. ie: Closet Lamp ( you choose the address )
- Type in the House and Unit code of the device : (address)  
Since it is a lamp, make it 'dimmable'
- Click OK and see the the unit appears in the Master List.
- Choose New Unit again and :
- Type in the Name "Closet Motion" ( you choose the address )
- Type in the House and Unit code of the device : (address)  
Since it is a motion sensor, choose to type in a simple script which makes other things happen when it 'goes on or off'...
- Click the ON script EDIT button:
- Type in the ON script for "Closet Motion" :  
  
turnon "Closet Lamp"
- Click SAVE
- Click the OFF script EDIT button
- Type in the OFF script for "Closet Motion" :  
turnoff "Closet Lamp"
- Click SAVE  
(yes, that's all for now)

Now, click OK, to finish, and find **Closet Motion** in the Master List. Notice that the Closet Motion unit has two **script icons (up and down arrows)** indicating that there is both an ON and an OFF script attached to this unit.

- Double-click on "**Closet Motion**" in the Master List, and watch the "**Closet Lamp**" turn ON.
- Double-click it again, and watch the "**Closet Lamp**" turn OFF.



- These events are recorded in the LOG window ! LOOK !
- Note also what happens if you single click on the little icon to the left of the unit name... see the pop-out command panel ?

### Second example:

- With the CONTROL key held down, Click select the "**Closet Motion**" unit in the Master List.
- See the pop-out panel for editing the Unit.  
Choose the Edit ON script
- Change the script to :  
if status of "Closet Lamp" is false then  
    turnon "Closet Lamp" for 2 \* minutes  
end if
- Click SAVE
- Click select the "**Closet Motion**" unit in the Master List.
- Hit the Return Key. This brings up the Edit Unit dialog again.
- Delete the OFF script. Click OK ...
- Now when you turn on the "**Closet Motion**", XTension automatically schedules an event to turn the '**Closet Lamp**' OFF in 2 minutes.
- If you pull down the Windows Menu and select "Scheduled Events", you will see an event which will turn off the '**Closet Lamp**' in 2 minutes.

### What's the difference ?:

In the first example, the motion sensor script turned on the lamp, and only when the motion sensor goes off is the closet lamp turned off.

In the second example, we used another XTension option and needed only one script which turns on the lamp for 2 minutes.

In each case, the result is the same. It just depends on how you want to write your scripts, and how your motion sensors behave. Note that some motion sensors do not automatically send an OFF.

### Other device types:

If the Closet Lamp is left ON for too long, it isn't any big deal. But if the Coffee Pot is left on for 10-12 hours, it can be a big mess.

So, you might want to attach a script to the Coffee Pot, so that any time you turn it on, it will always be turned off automatically within a certain time.

In this case, you would want to attach a script directly to the '**Coffee Pot**' which simply says : turn off (thisUnit) in 90 \* minutes

These subtle things are the reason why we have so many XTension verbs.

And it is why we provide the scripting and scheduled events system where you can schedule an event for 10AM or 3AM etc which routinely goes around and commands all your devices into their default states...sort of a garbage collector....

You might choose to have a script called "Spring Cleaning" which turns on all of the lights in your hidden places and keeps them on until you either manually command it off, or the 'garbage collector' comes around...

Putting this all together, you might easily create a very hospitable system which does things like :

If it's a weekday and the "shower motion" indicates that there's a live body in the shower.

Turn off the "bedroom alarm", turn on the radio, and bring up the lights in the house

if it's still dark outside...

Don't forget that you can also use all of the facilities of your Mac :

Voice annunciation, speech recognition, sound files, video capture, website upload .....

Eventually, you should come back and look further in the Manual.....

Don't forget when you want to turn on everything for real, you must Set Serial Comms ON, and turn Monitor mode OFF before you can actually issue commands....

If you can at all get to the Internet, then Please visit our site at  
( [www.shed.com](http://www.shed.com))



# Take a test drive

## Start up XTension with the example database.

Because the OS X version of XTension is serialized in the database rather than the application, there is a serialized DEMO version on the downloads page all the time that is good for about a month.

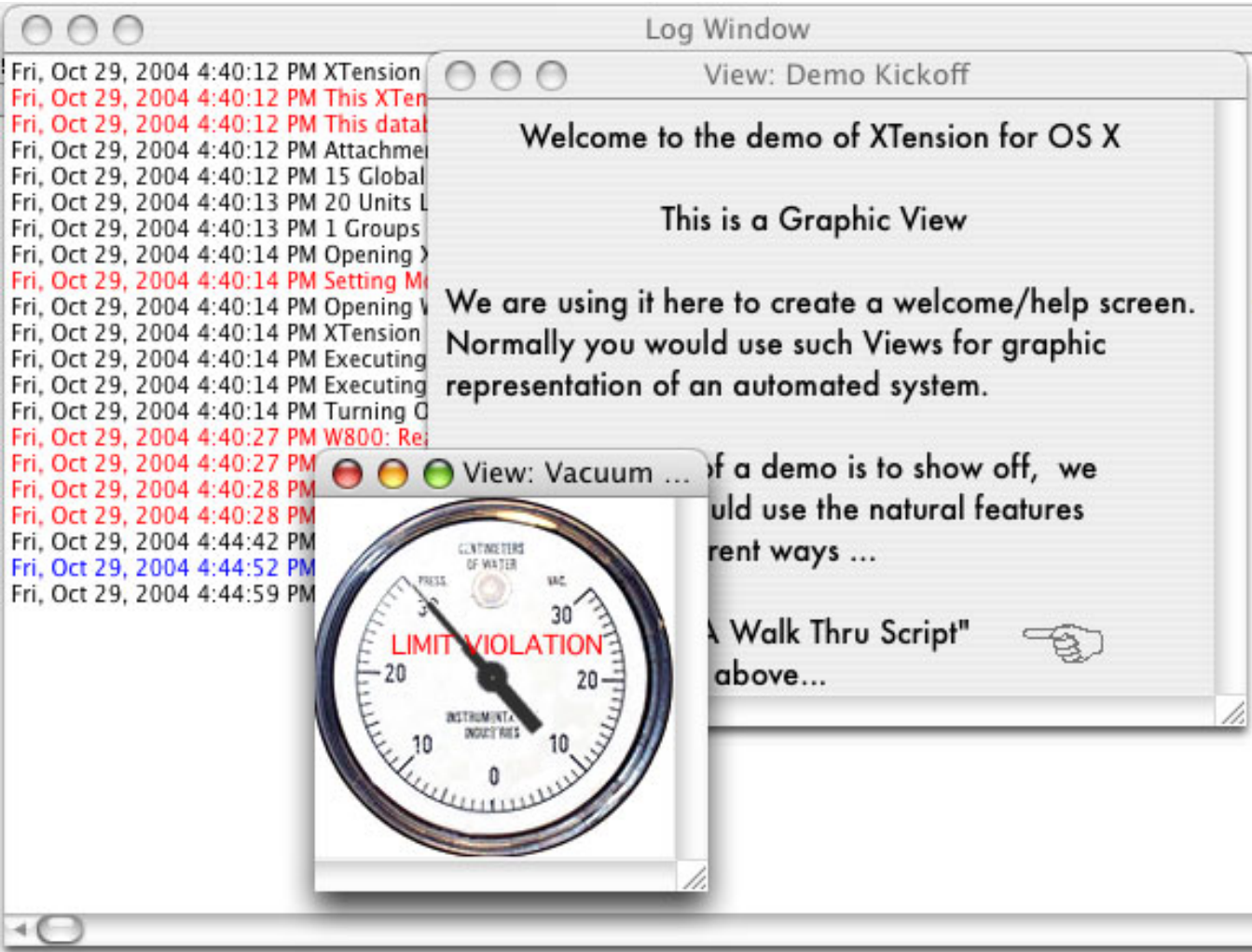
You can get this at :  
<<http://www.shed.com/xtension/X4XDEMO.dmg>>

Download that image and follow the instructions to place it on your desktop etc.

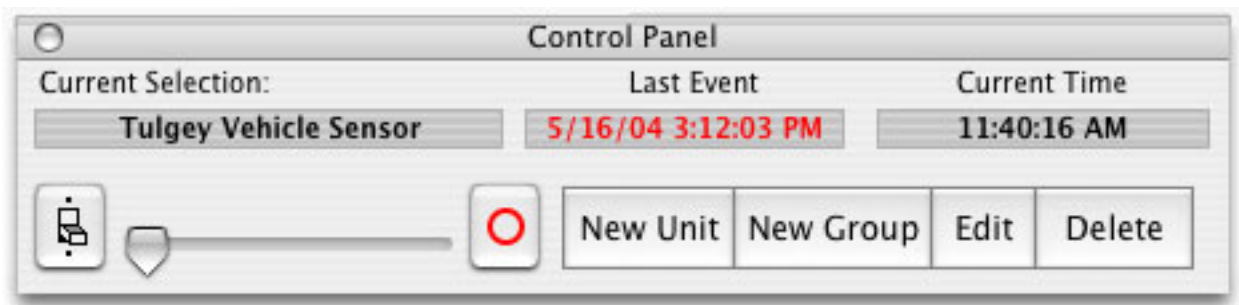
Double click on the XTension program icon. The program will start up with several windows and an automatic demonstration.

When you've gone thru this once, you'll have a better idea of what kind of things to expect from the software.

After you're sated with the demos, the following pages will make more sense.



## The Control Panel



This window 'floats' above all other XTension windows when it is present.

The left part of this panel always shows the state of the currently selected unit. If no unit is selected in any other window, the left part of this panel will be un-interesting.

Note that if any unit is selected in any View or List, then the "Last Event" time will be for that selected unit. If there is no unit selected, then this time is the last event time of any type.

From this part of the panel you can change the state of any unit by the **ON/OFF** switch, or if the unit is 'dimnable', you can set the level with the **slider** switch. If the slider is 'grayed' out, or un-responsive, it means that the unit is not 'dimnable'.

If XTension indicates that the unit is already ON and you want to send another ON command : *Shift-Click the toggle switch !*

This is slightly different when the selected unit is a 'simulated preset dim' type of unit. A shift-clicked ON will get more BRI commands, and a shift-click OFF will get more DIM commands. The number of dim/bri steps should correspond to the current 'preset level'.

The icon to the right of the slider is the '**Blocking**' control. It displays the 'Blocked/Unblocked' status of the currently selected unit. ( see the Blocking chapter for more )

The right hand side of the panel shows the current time and the time of the last event. (XTension uses the Macintosh system clock.)

Below these are 4 icons which allow you to add a new unit, create a new 'group', choose items to be in a list or view, edit the current item, or delete the selected item.

The **New Unit** button will prompt you to add a new unit to the database, and the **New Group** button will let you create a new named group of other database items.

The **Choose** button is used in several cases to choose or select the items in a List or View.

The **Edit** button will bring up the Edit Unit or Group dialogs. They will only work when you have a current unit selected in one of the Views or Lists.

The **Delete** button will delete the selected item from the database if you are viewing the Master List. Otherwise, it will simply remove the selected item from the front window, whether it's a List or a View.

( If you just want to remove an item from a list or view other than the Master List, then just click-select the unit and press the **Delete** key. )

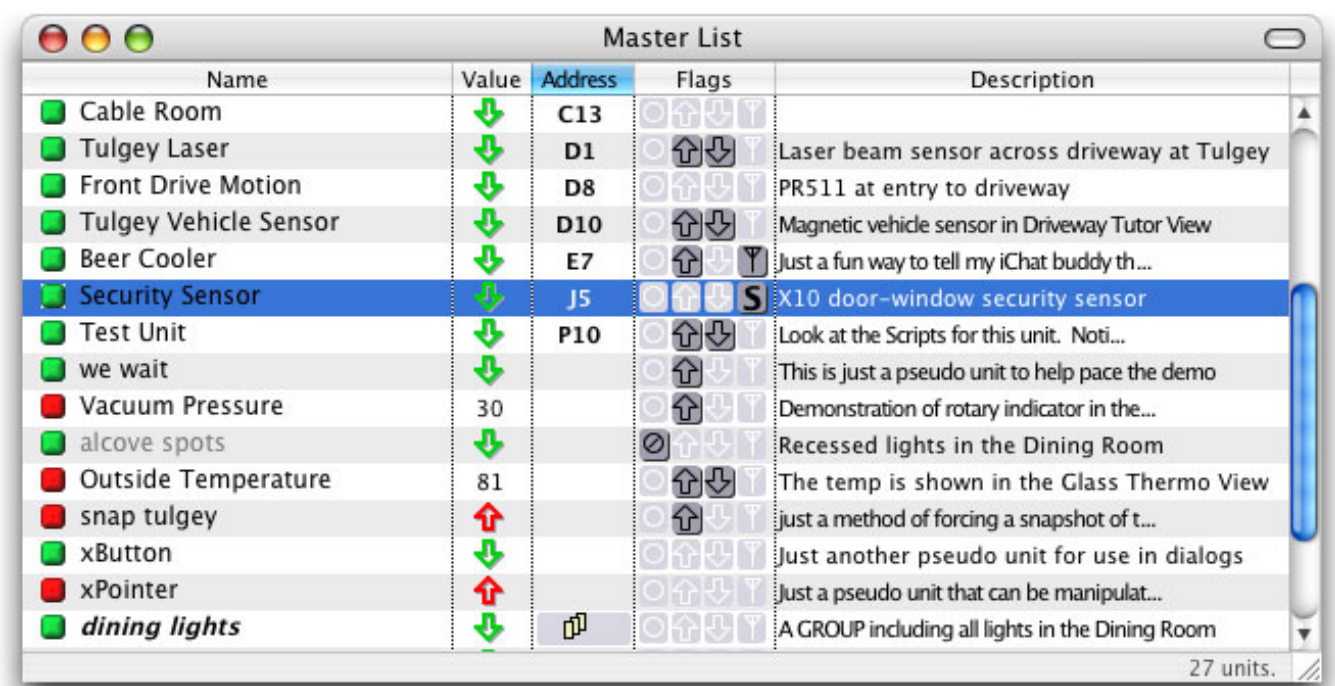


# The Master List Window

This window contains a list of all of the units in the database. From this window you can create or edit units, and control or watch their behavior.

You can send this window away, but you cannot delete it permanently. It can always be retrieved via the 'Windows' menu.

The order and presence of columns, as well as the sequence of items, is completely user-directed.  
( see the next chapter )



The **status** of each unit is indicated by the small icon at the left of the name.

Units which can only be on or off are indicated by the 'up/down arrow' icon in the Value column, and **Dimmable** units are represented by a number which corresponds to the **current value** of the unit. Lamp modules will range in value from zero to 100. 'Pseudo' units can vary from (a very large negative number) to (a very large positive number -- 64 bits)

If the unit has an X-10 **address**, it is shown. Note that some devices do not have an address. Groups do not have addresses, 'pseudo' units as well do not need them. An icon that looks like three documents in the address column indicates that the unit is a **group**.

The presence of a blocked-circle icon in the status column indicates that the unit is **'blocked'**. You can manually remove the 'blocked' status via the Control Panel, but until you do, you will not be able to change the state of a unit from within XTension.

The **up/down arrows** indicate whether you have created ON or OFF **scripts** for this unit.

The icon that looks most like an **antenna** is the RF-Allowed indicator. If this icon is bold, then this unit is controlled by a standard X10 **wireless** remote or sensor.

The icon that is a BOLD letter "S", indicates that this unit is a X10 **wireless** "Security Module".

The **description** column is the text description given to the unit in the edit unit dialog. It is for your use, not required by XTension. Do notice that there are AppleScript verbs that can be used to dynamically change the contents of the description field. Note that the Description field can be directly edited by clicking the cursor in that area and typing...

A unit that is a **Group** will be shown in **bold/italic** font, and it will not have an address and cannot be 'RF'.

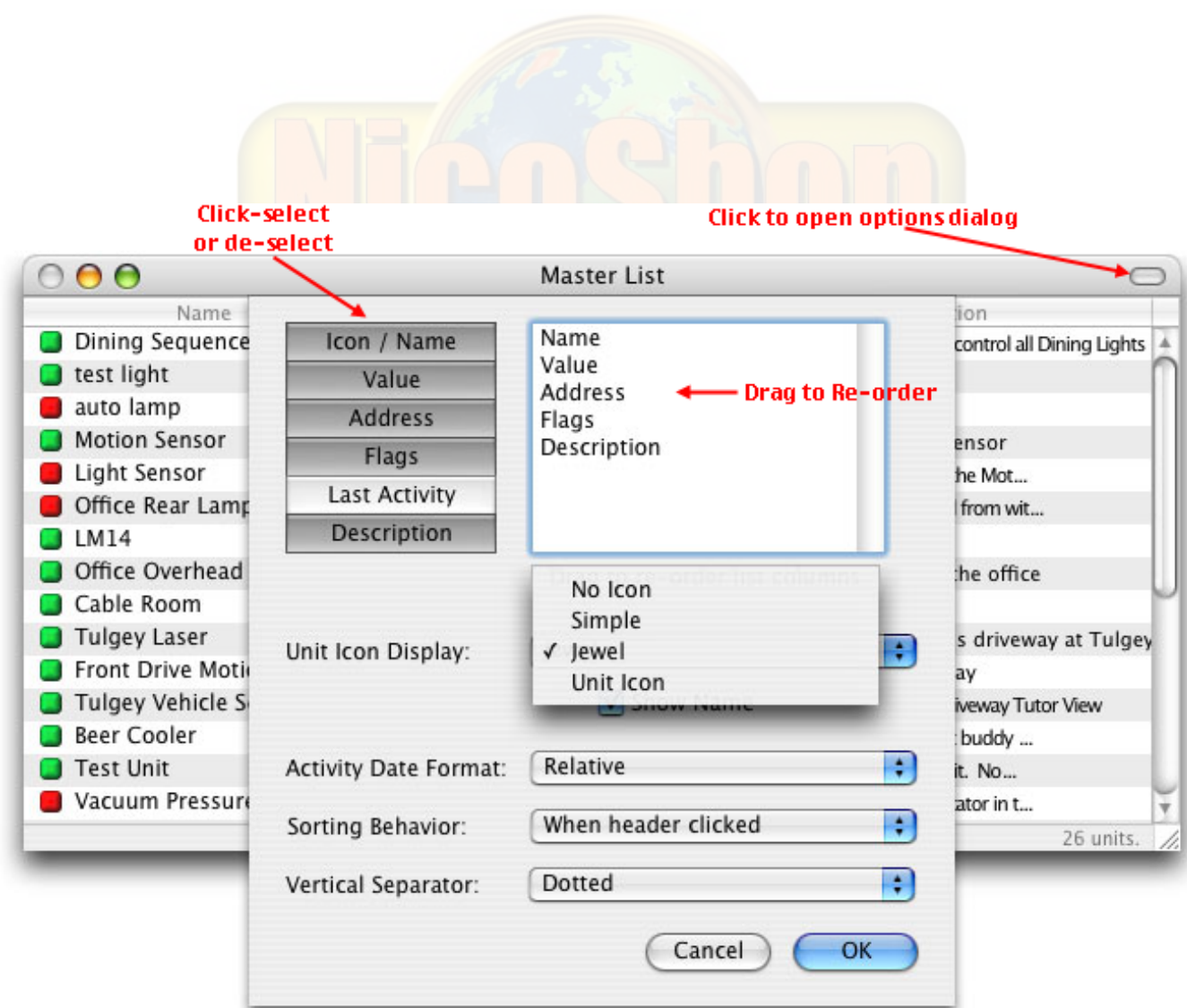
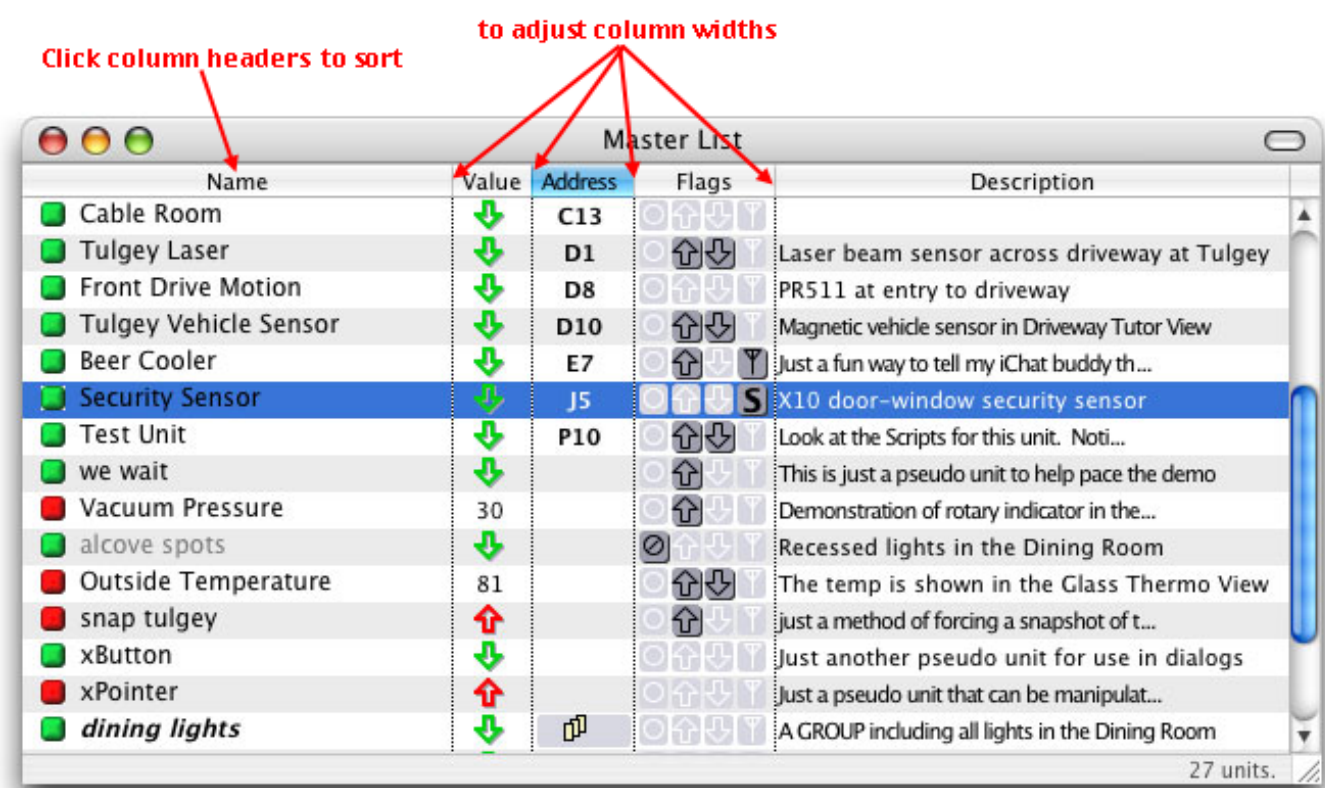
Point the cursor at the item “**Office Rear Lamp**” and hit the Enter key. You will see the **Edit Unit** dialog window.





# List Window Display Options

The graphic display of all of the list windows can be individually configured !



To choose which columns you want to display, select or de-select the column name in the left hand panel of the dialog.

To choose the order of display, drag-re-order the items in the right hand panel.

### **Unit Icon Display**

The pop-down allows you to choose the type of icon for each unit.

This icon is the one on the left hand portion of the Name field.

Notice that the "Unit Icon" option will display a scaled-down image of each chosen unit icon.

If no icon is chosen for the unit, nothing will be displayed.

### **Activity Date Format**

This option is for the "Last Activity" date format, and is only seen if that field is chosen for display.

### **Sorting Behavior**

This option determines how or whether the list is sorted.

**None** = No sorting will be done automatically, and none when the headings are clicked.

(this means that you drag-order the items, and they stay that way regardless.)

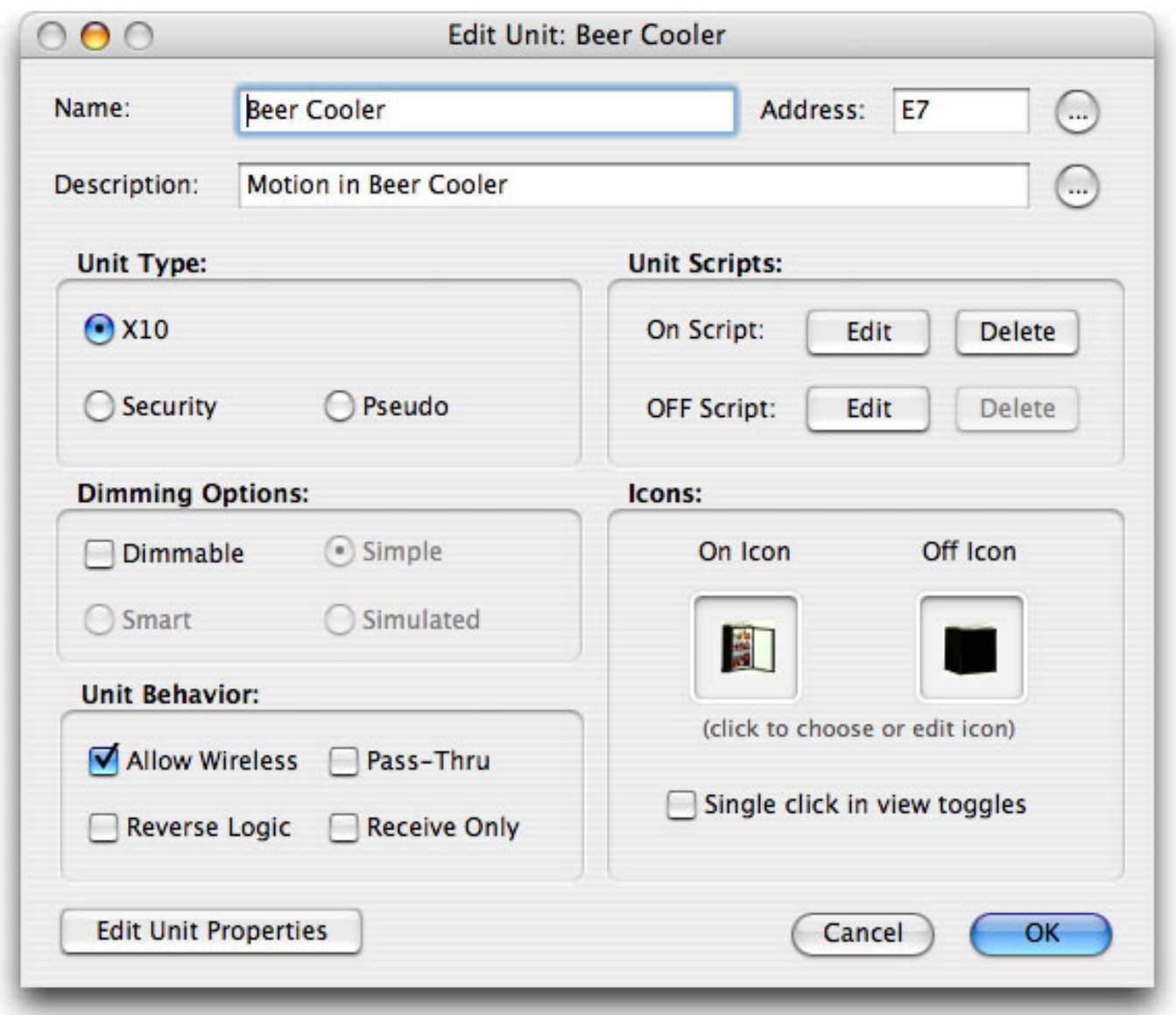
**When header clicked** = Normal mode. Click a column header and the rows will be sorted either in ascending or descending order.

**Automatically** = Items are re-sorted every time the Value of any item changes. Best used for watching groups of units which have similar Values.

**Vertical Separator** : Just a nice option to allow you to separate items to suit your eyes.

# The Edit Unit Dialog

This dialog is used to change all attributes of any unit except for its current state. Whenever you add a new unit or change a unit, you will see this dialog :



Here is where you associate an X-10 **address** with a unit **name**, assign **icons** which appear in graphic **views**, select special **options**, elect to assign **scripts** to state changes and even create ad hoc unit **properties**.

The **Name** is any alpha/numeric string up to 28 characters in length. It may contain blanks, and under\_score, but may **not** contain any quotation marks, colons, or slashes.

The **Unit Type** is a pop-down selector. Notice that this is where you determine which major type of unit this will be.

**X10**.....is obviously a 'real' X10 unit, either powerline based or wireless.

**Security** is a special class of wireless which indicates that this unit is a special X10 Security module.

**Pseudo**....is a unit that is used entirely as a counter or flag, having no 'real' physical counterpart.

Note that if a unit is type **X10**, or **Wireless**, you must assign it an Address (see Unit Options panel).

The **Address** is any valid X-10 address, with house code between A and P and unit code between 1 and 16.

If the unit is type **Security**, that device has a set-up procedure where it will

randomly choose an address and report it. Please see the **Wireless Security Units** chapter for help in setting up such units.

## The Unit Scripts :

The **ON and OFF script buttons** allow you to assign scripts, edit an existing one, or remove either of them. When they are greyed out, they haven't been created.

Push the ON script button for "Beer Cooler". You should see the Edit Script dialog:

( chapter after **groups** ) >>

## Dimming Options :

**Dimmable** -- This check box indicates that this unit will accept X-10 'Dim' commands, or is a 'Pseudo' unit (above). Unless this box is checked, the slider switch in the Control Panel will be grayed-out.

**Simple** -- this option is for the simplest X10 dimmers, or 'pseudos'.

**Smart** -- this option is for more modern Dimmers that offer 'internal' preset dim options. This simply tells XTension how to expect the unit to perform, and thus 'track' the behavior of the unit in response to commands to it.

**Simulated (Pre-set Dim)** -- this is a specially nice feature of XTension. Whenever you set up a new lamp module (standard X-10 issue), make it 'Dimmable', and you will get this option. Choose this option, and thereafter, XTension will control the behavior of the lamp much as if it were a 'smart' (and much more expensive) dimmer.

(Please see the chapter "Simulated Preset Dim" for more.)

## Unit Behavior :

**Allow Wireless** -- If selected, XTension will recognize RF commands for this unit.

**Pass-Thru** -- this option is available only for 'wireless' units. If selected, whenever a ON/OFF/DIM/BRI command comes in from the MR26/W800, that command will automatically be passed-thru to the powerline, **before** the Unit Script is executed.

**Reverse Logic** -- Some devices such as laser beam motion sensors send an OFF command when they are triggered, and an ON when they reset. This check box tells XTension to reverse this logic so that scripts can be written using 'positive logic'.

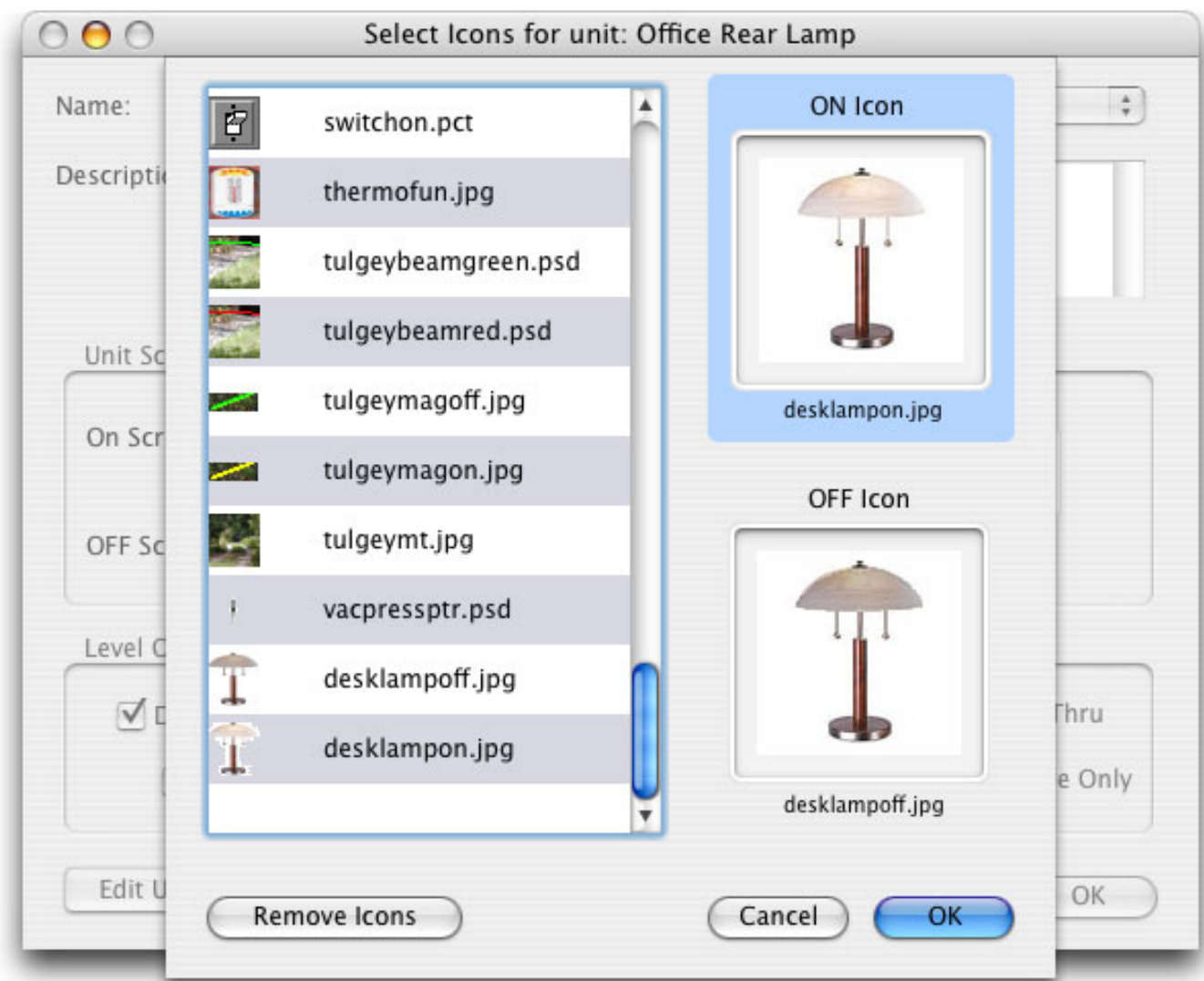
**Receive Only** -- This check box indicates that the unit should not be controlled by any script. XTension will record any changes received from the X-10 controller, but will not allow any script to issue commands to this unit.

## Icons

The icons that you select are those that will appear when you add this unit to any graphic **view**.

The **On and Off Icons** are not 'paired'. This means that you may choose an icon for the ON state which is totally different from that of the OFF state.





You add icons to your system by creating them with a graphic editor, and putting them into a folder named **Icons** in the same folder as the XTension application.

Just click on either the ON or OFF icon and the selection dialog above appears. Find the icon you want, and click on it. That will appear in the panel on the right. Click OK, and from then on, you can simply drag a unit from any List window, onto any Graphic View window, and that icon will appear. Don't forget ...it is HOT... you can click on it in the graphic view to change it's state.

**Single Click in View Toggles** -- This check box indicates that you want to be able to click ONCE on the unit icon in a **graphic view**, and make the unit change state...from ON to OFF and vice versa. This eliminates the need to double-click. This is very handy in some cases.

## **Wireless Security Units -- W800RF32 only**

**Available only if you have a W800RF32 wireless receiver**

**For many years, X10 has offered a line of security (burglar alarm) systems that employ wireless sensors that are completely incompatible with the RF transceivers of their 'home automation' products.**

With the advent of WGL Designs' W800RF32 wireless receiver, it is possible to receive both the traditional wireless devices and the 'security' devices.

There is one major difference between the two types.

The 'security' devices cannot be set to a specific address like the home automation devices.

The security devices always select a 'random' address, when they are first powered up, or when you change the batteries, or 'reset' them.

This causes a bit of a bother in that 'collisions' may occur as you increase the number of these devices in your home.

In order to relieve the confusion and crowding of the address space, XTension automatically maps the addresses of the wireless units into their own separate address space.

XTension can always distinguish the difference between messages from normal wireless X10 devices and those from wireless 'security devices'. Thus you never need to worry about a normal wireless device and a security device being confused just because they have the same 'house/unit' address.

### **Setting up a new security unit**

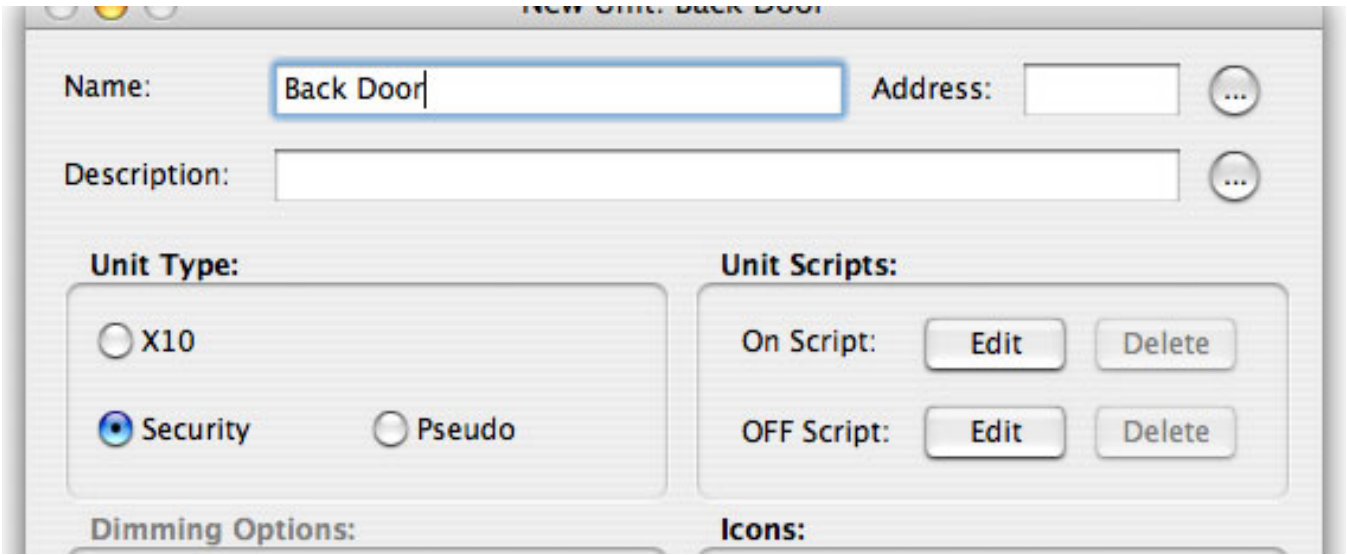
You should indeed read the instructions that come with the device, but basically all you need to do is install the batteries while XTension is running, (and it doesn't hurt to have the device near the W800 as you do this.)



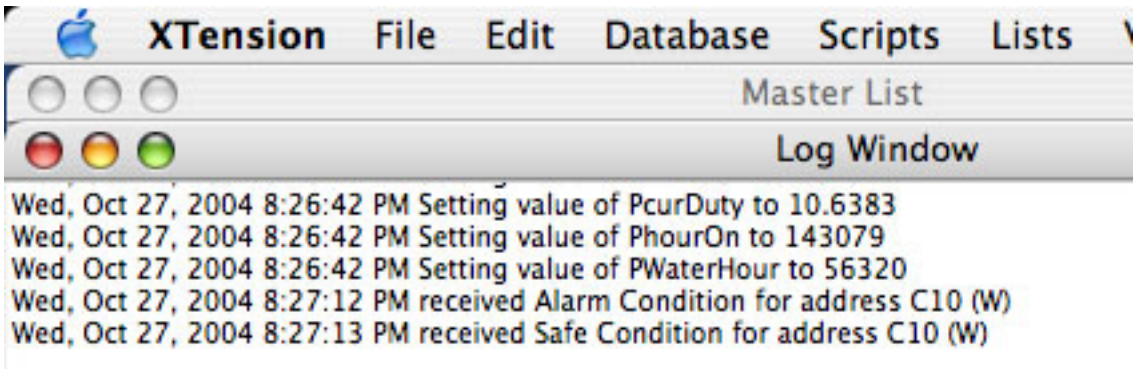
Now tell XTension that you want to create a New Unit...

Give it a Name... and click on the **Security** button in the Unit Type...





Now Push that TEST button on the unit and see that the Log Window shows something like this :

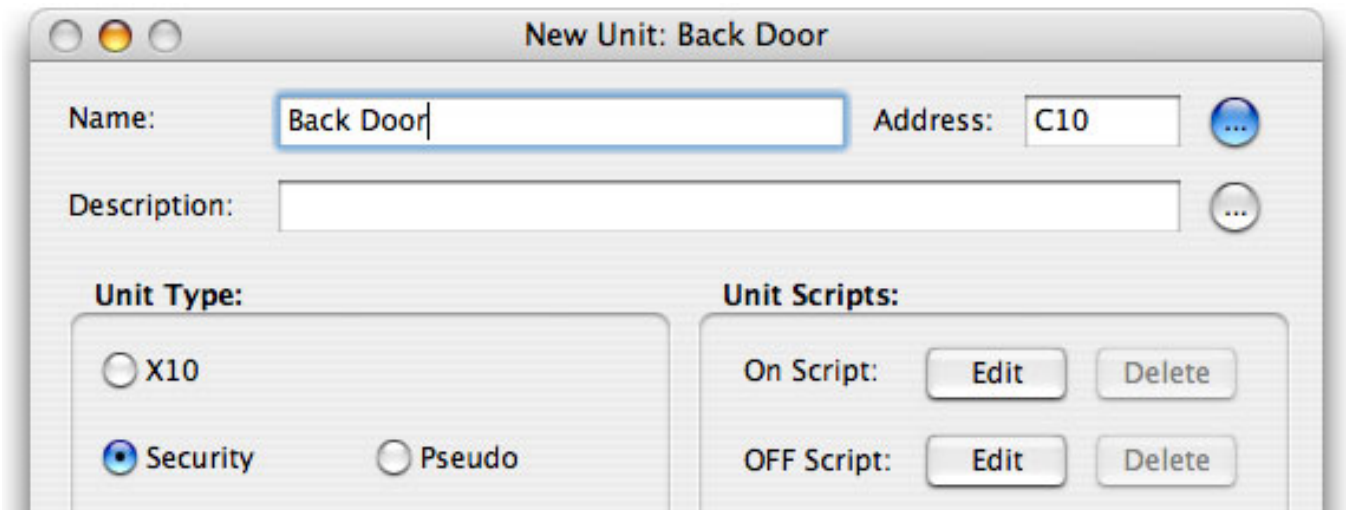


After you've completed the configuration of the new security unit, you will see the name of the unit in the Log rather than the address.

Notice the distinctive difference in the log entry for an ON versus an OFF event...

Next, click on that "•••" button in the upper right corner...

Notice that the Address is filled in automatically by XTension from the last 'un-identified' address received by the W800...

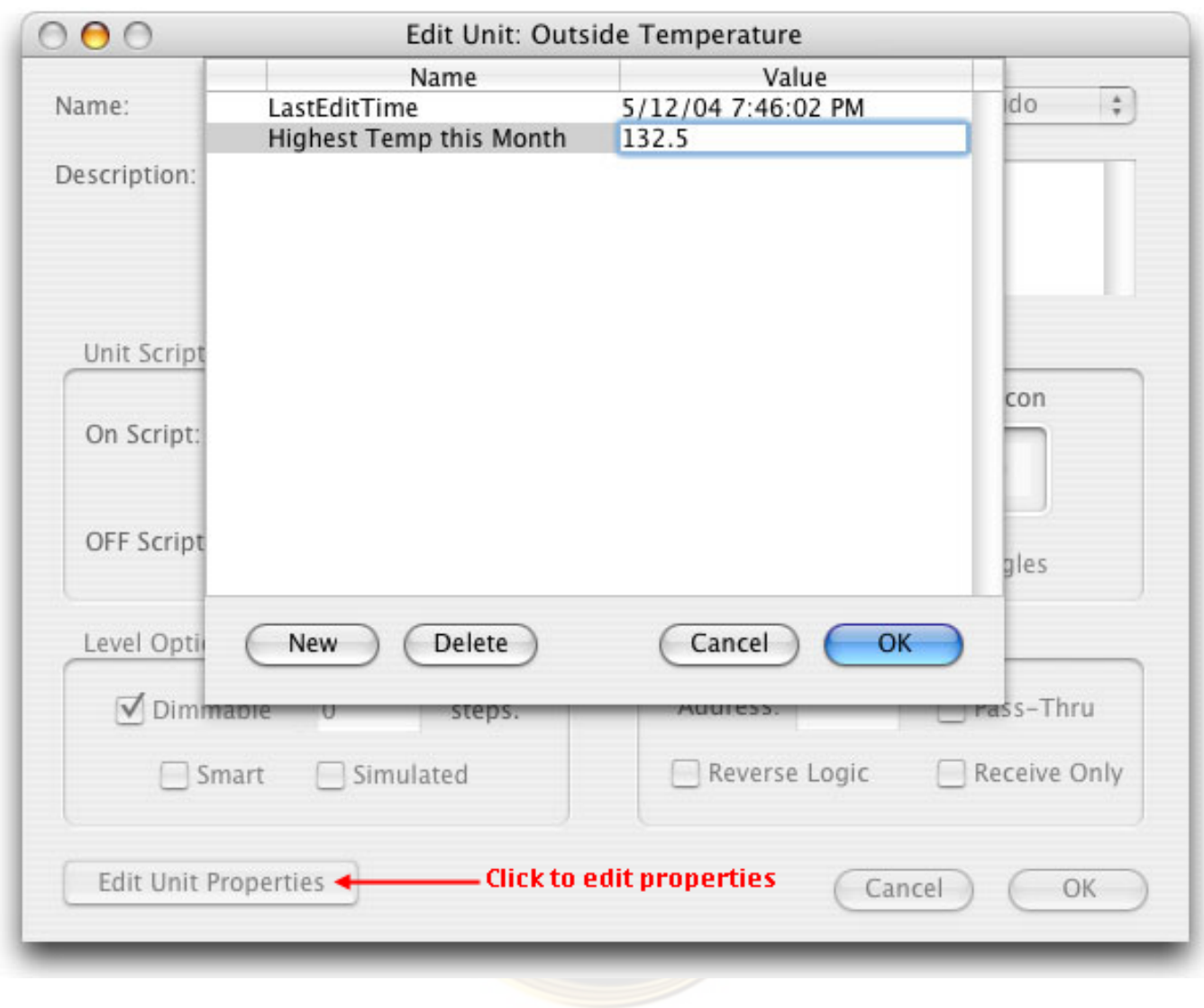


Notice that 'security units' cannot be 'dimmable', cannot be 'passed-thru'.

# Ad Hoc Unit Properties

Every database unit may have unique properties that are associated with that unit and no other.  
You may choose to create these properties according to need, just as you might create '**pseudo**' units in previous versions of XTension.

These 'properties' are named, have value, and will be saved with the unit record whenever it is saved.



The **name** is restricted similarly to the Unit name.

The **value** is always stored as a 'string', which may always be manipulated by AppleScript as a numeric or string value.

Notice that there is always one unit property **LastEditTime** attached to every unit.  
This can be very useful in finding units that were recently edited...

## Verbs that act on unit properties

There are of course some intrinsic XTension verbs that let you get and set these unit properties :

### set unit property : set a named property to a value

**set unit property** string (existing property name)  
to string (any string value)  
[ in unit string ] The unit name into which to set the variable. Optional. If omitted,  
the assumed unit is (thisUnit)

**get unit property : get the value of a named property**

**get unit property** string (existing property name)

[ from unit string ] The unit name from which to get the variable. Optional.  
If omitted,  
the assumed unit is (thisUnit)

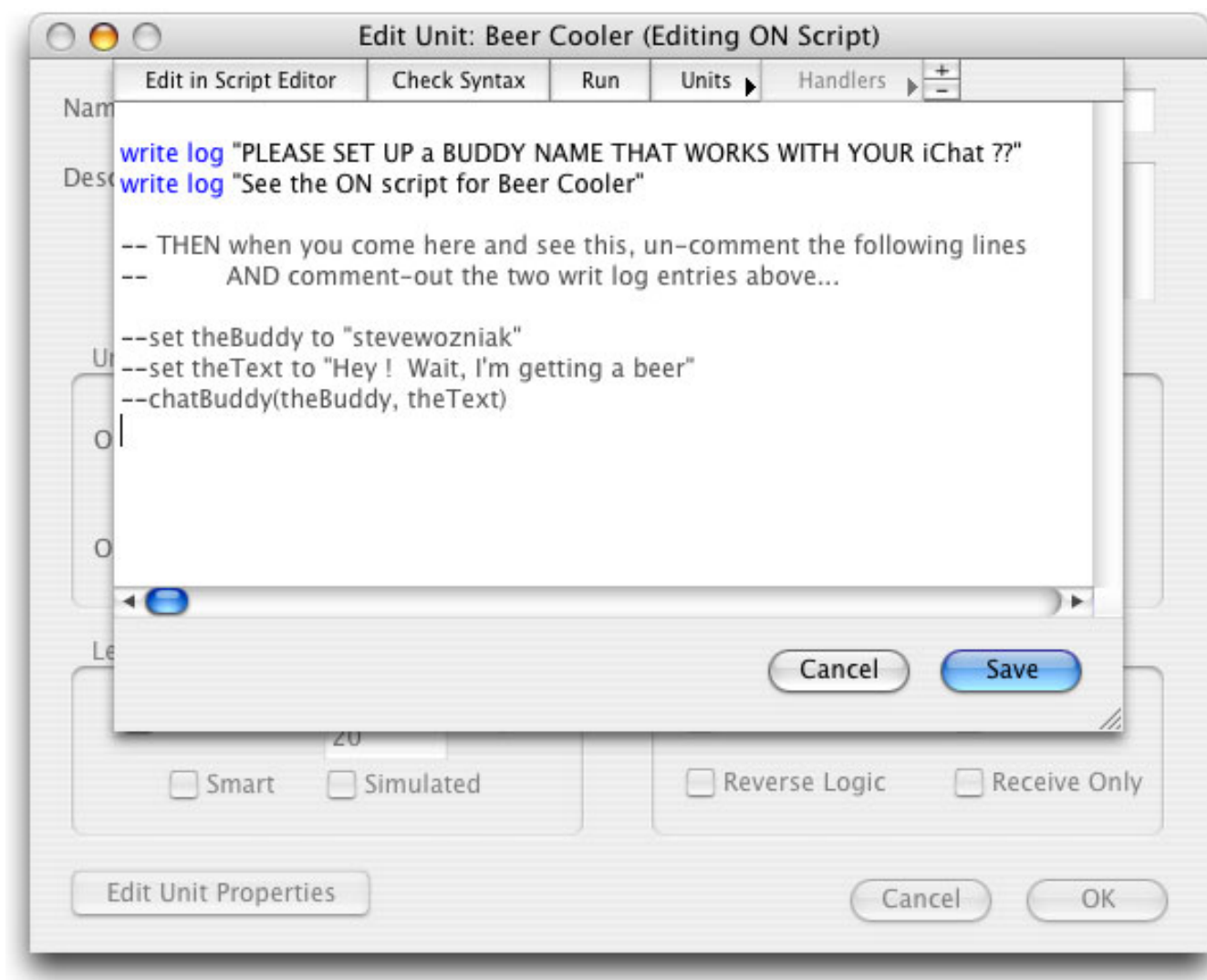


## Edit Unit Scripts

While in the Edit Unit Dialog, press either the ON or the OFF script EDIT buttons...

This dialog is used to change all attributes of any unit except for its current state. Whenever you add a new unit or change a unit, you will see this same dialog.

Note also that this dialog can be brought up by selecting a unit in any list, and pressing CONTROL, and then selecting one of the options from the pop-up menu.



**Notice that the 'context' of this script is that the Unit has been commanded to change to this state, either from some internal event, or some external event.**

**The new 'state' is either ON, (or value is not zero),  
or OFF ( or value is zero ).**

**The unit OFF script will only be triggered, when the 'future value' of this unit is Zero or OFF.  
Please see the chapter "Reserved Variables" for more 'self-referencing' options.**

Any legal AppleScript verbs and constructs are valid here, as well as any of the hundred or so native XTension verbs.

Remember here that if the 'stimulus' is from some internal function, then the Command has not yet happened, and thus anything that you do in this script

will delay that command's effect.

If the stimulus is from an external event, then the 'event' has already occurred, and you must respond to it quickly and reasonably.

Please note that not all units need on/off scripts.

Things like **motion sensors** should have scripts so that you can 'react' to those events.

Things like **lamps** seldom need Unit scripts, since they most often are being commanded by some person, some internal script, or scheduled event...

Lamps like the 'closet' lamp, which seems to always get left on, or the coffee pot or hair curlers that simply CANNOT be left on, could have Unit ON scripts that can automatically schedule the turn OFF of the device after some reasonable period.

SMART switches and dimmers can announce changes that occur 'manually' at the switch. Thus these 'events' are interesting both from the idea of keeping the database up to date, and the idea that such switches can represent input to an 'alarm' system... !



# The Edit Group Dialog

By creating **groups** of units, you can control several devices with a single XTension script or button.

The **DEMO database** includes a group named “**dining lights**” which refers to all of the commands for the dining room lights of the example house.

**Any** command to the **group name** will cause XTension to issue the same command to **each** of the members of the group individually. Thus you can turn them all ON or OFF with a single :

**turnon** “dining lights”

OR:

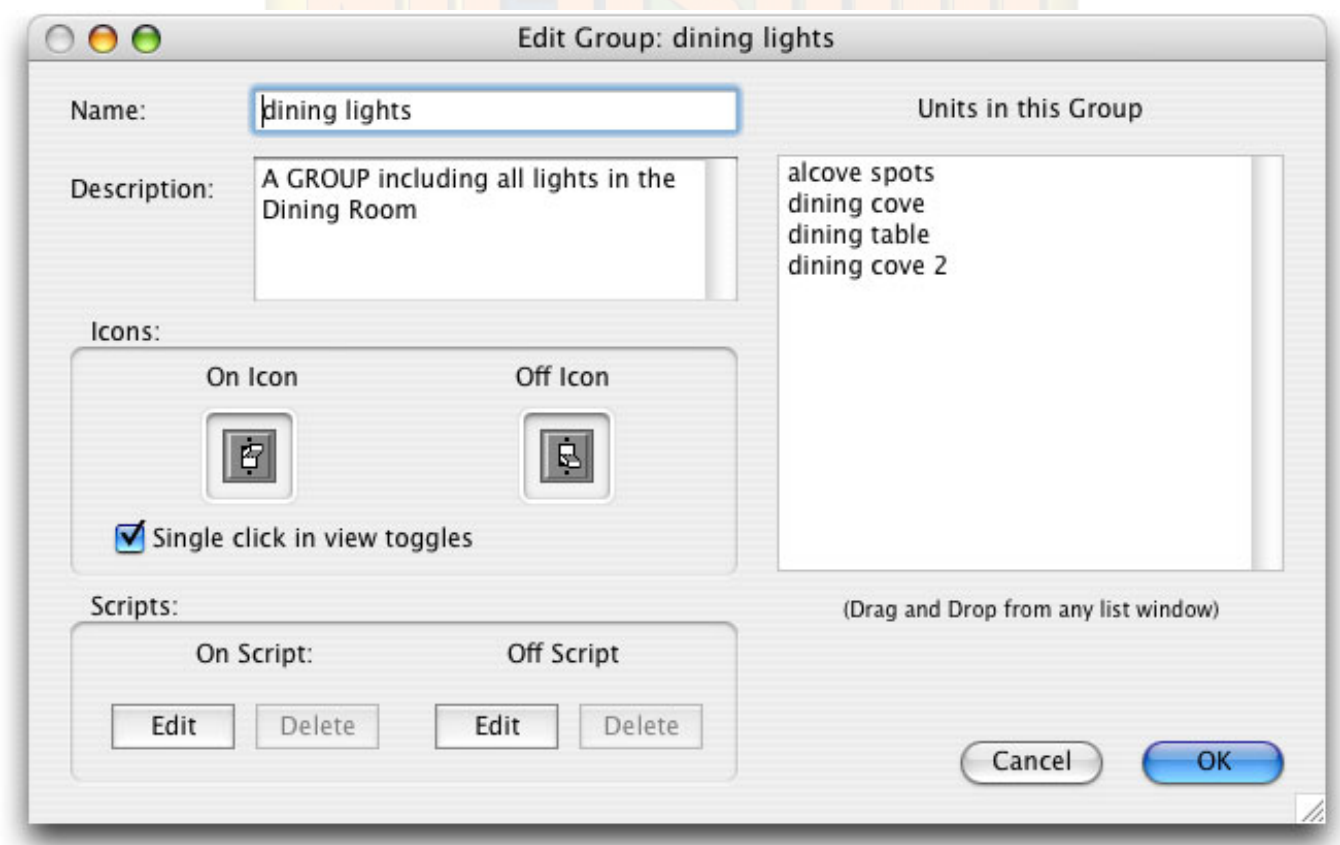
**dim** “dining lights” to 75

Groups are also useful when you want to create a List of items of special interest, such as "All Motion Sensors".

For example, when you are modifying the scripts/parameters of a group of related units, you may want to create a limited List window in order to reduce the amount of scrolling you might have to do from the Master List window.

**The sequential order of the units in any Group is important:**

It is possible to 'drag around' the units which are in a Group, thus 'ordering' them such that **you** can specify the order in which they will be 'commanded' whenever you issue a command to the entire 'group'.



Note that you can add a **group** to this **group** (yes you can...)



XTension will even attempt to tell you whenever you try to add a group to this group that is then indirectly linked to 'this group'.... a looping situation ... that's not a good thing... :-)



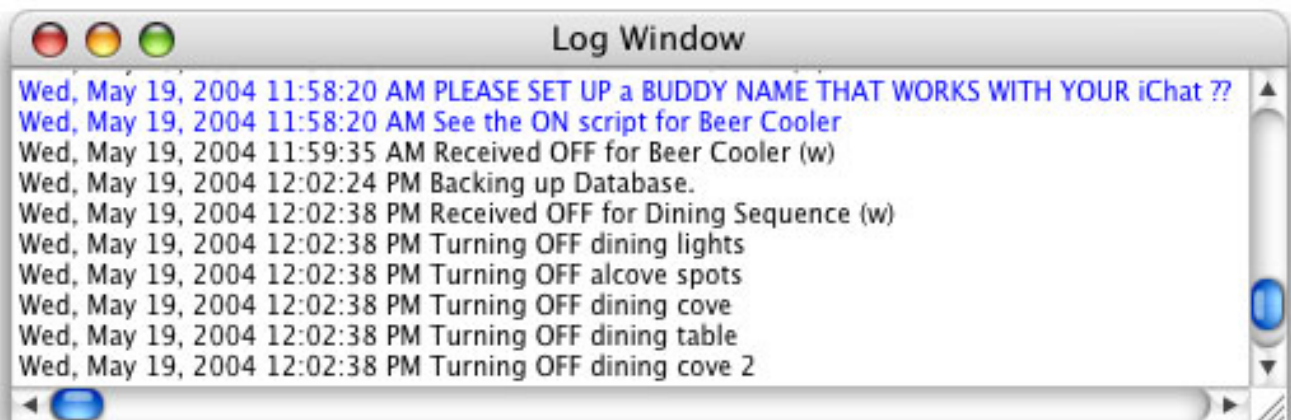
## The Log Window

Each of the changes that happen within XTension are recorded on the log. You can even write your own messages to the log.

**You** can change the [level of detail](#) which is displayed in the Log Window. See Preferences/Logging in the Edit Menu, and the AppleScript verbs which allow dynamic changes !.

**Please** do not ignore the value of this feature. In this window you will see timestamped messages regarding errors, explicit messages, and normal activity.

\*\*\*\*\*



There are four 'colors' used for items in the log :

**Black** is used for all low level, non-exceptional events. ••

**Blue** is used for any item which is written by a script to the log.

**Green** is only used by explicit User write log commands.

**Red** is used to indicate any error or operational exception.

•• **NOTE:** **Black** is generally meant to signify 'normal' events.

Be sure to check out the 'write log' verb for the ability to set the color of text that you wish to write into the log.

If you are not editing your system, keep the Log window present. In normal operation, you **do** want to have it readily available.

Regardless of what you do, you cannot prevent all of the log data from being written to the Log File on disk...under normal conditions.

It is annoying that while you are scrolling back in the log to examine previous events, events continue to occur and that causes the Log window to scroll automatically to the end. Recent versions allow you to scroll back in the log and not worry that new events will cause a re-position.

Do pay attention to the scroll bar indicators as you would in any other such window.

# The Scheduled Events Window

This window shows all currently scheduled events, and allows you to create new ones. Scheduled events may be one-time or repeating. You can sort the order, and you can suspend events.



The icons at the top of the window allow you to create **new** events, **delete** existing ones, or **sort** the items either by time of day or by the event names.

The **New** event button will bring up the Edit Event dialog and prompt you to create a new scheduled event.

The **Edit** button will allow you to change any scheduled event.

The **Delete** button will permanently remove a selected event from the list.

The **Suspend** button will allow you to suspend any scheduled event without removing it from the list. Selecting the event and pushing the suspend button again will un-suspend the event.

You can suspend and unsuspend an event manually or via XTension script verbs ( see ).

The **Name** and **Time** buttons will allow you to sort the events in the list either by their names or by the time of day of the event.

The icons in each event panel indicate whether the event is **repetitive**, or **1-time**, and whether the event turns a unit on or off, or executes a global script. Along with these is the name of the unit or script associated with the event.

The **R** icon indicates whether the event is repetitive, and the **up or down** arrow indicates that the named unit will be turned on or off. A **as** icon will appear for events which execute a script.

If the event has been **suspended**, a familiar 'Stop' icon is shown.

The time and date that appear are the time/date when the event will occur. If the event is repetitive, this is the next time that the event will occur.

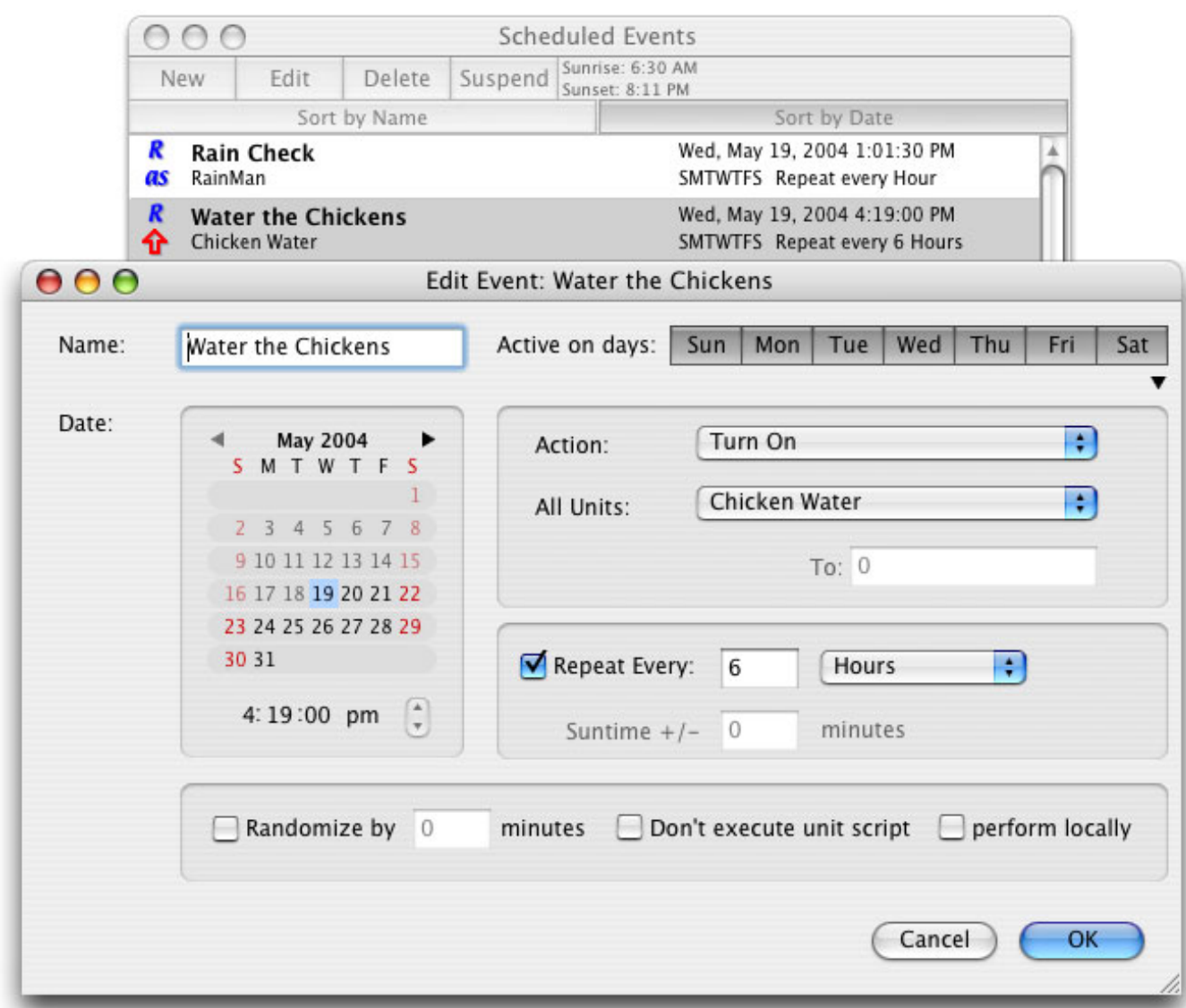
Be sure to see the create/remove/suspend event-scripting verbs.

If you double-click on the '**Water the Chickens**' event, you will see the Edit Event Dialog : (next page)



# The Edit Event Dialog

This dialog is presented when you push the 'New' button in the Scheduled Events window, or double-click/edit an existing event.



With this dialog, you can create a Scheduled Event which is either 'once only' or repetitive. You can elect to turn a unit on / off, dim a unit, or you can cause the execution of a global script. Valid unit or global script names appear in convenient pop-up menus.

First select a date and time, and then whether you want the event to **repeat**. On repeating events, you select the repeat period from a pop-up window of seconds, minutes, hours, days, weeks, months, or even 'suntime' relative. Note that you may also select which **days of the week** you wish to include or exclude from the schedule.

**Please Note:** It is not a good idea to create an event which repeats more frequently than once per minute unless that event requires a very short time to complete. If you create an event that takes longer to complete than the repeat time, it is possible to get into a 'lock up' state where nothing happens but that event.

An example would be a script that tests some condition of another application or an internet function which may require that the modem connection be established first.

You may also elect to **Randomize** the event by a number of minutes less than or greater than the specified time. This is useful when you wish to make your house look occupied, without looking like it's automatic.

Note that you must select an '**action**' to perform. This can be 'turn on', 'turn off', 'toggle' (flip the other way), or 'block/unblock' a unit. You can also '**execute a script**'.

XTension version 3.2 and after offer the action of '**Preset**',

which specifically sends the extended code sequence for setting the internal 'preset dim' levels of devices that will accept this command. See the 'extended codes' chapter.

You may also elect **not** to execute the unit script with the turning on or off of a unit.

**Note also:** The check box "Always perform locally" has no effect unless you are using the 'targeting' feature. (see Multiple Macs)

Once you click OK, the event will appear in the Scheduled Events window identified by the name that you gave in the event name field. If you do not enter a name for the event, XTension will create a name using the current time.

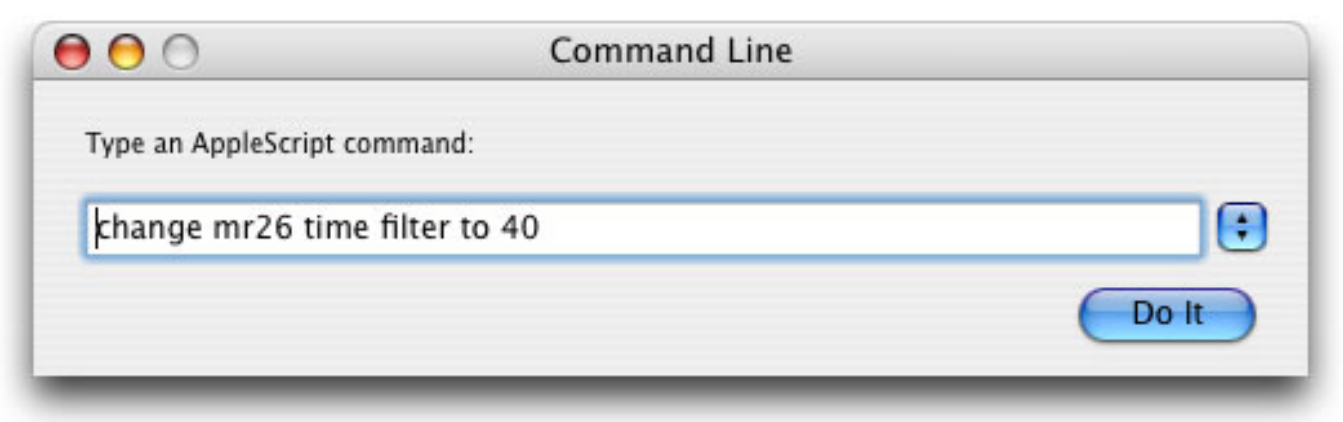
Now let's try a tool that can execute any AppleScript :





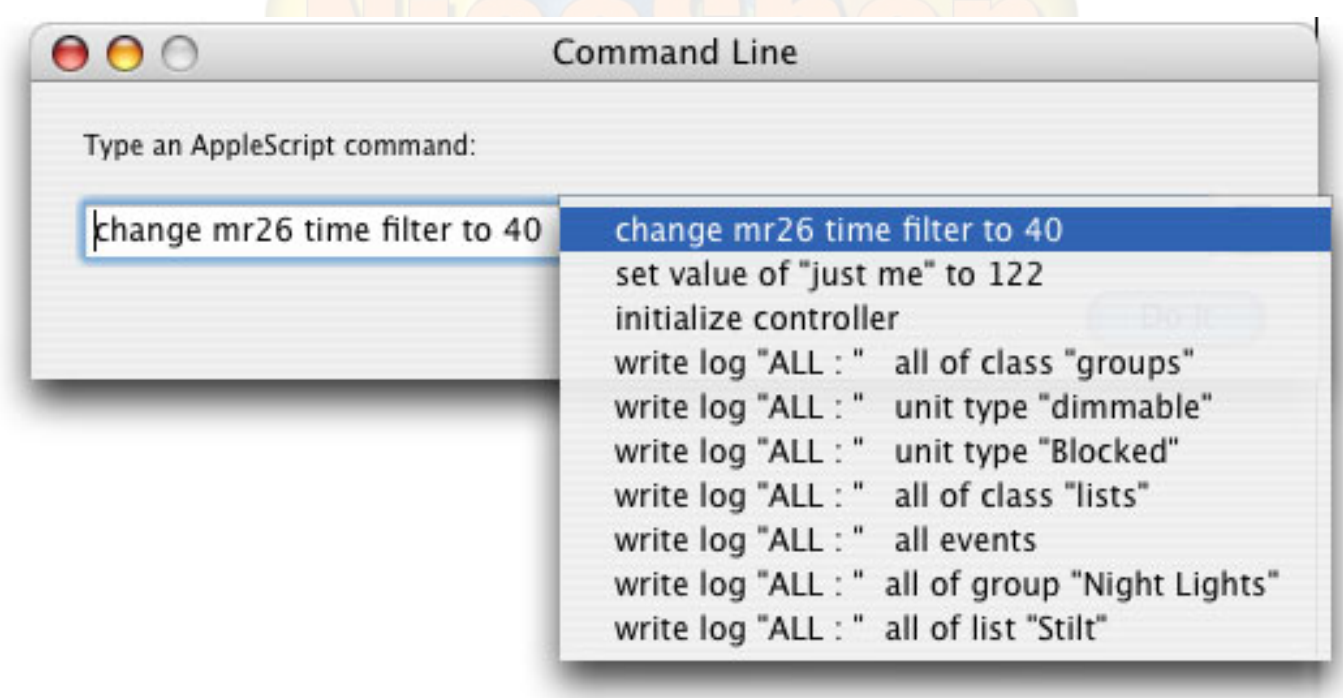
# The Command Line Window

The Command Line is included for your convenience. Any valid AppleScript command can be typed into this window and executed 'ad hoc'. This obviously includes any XTension phrases.



If you type in this command, with the example database, and then push 'Do It' , you will see an event message in the Log Window that announces the result.

In the OS X version, there is a new feature that remembers the last 10 commands that you typed into the command line... You can choose any of them from the pop-up, and just click 'Do It' ...



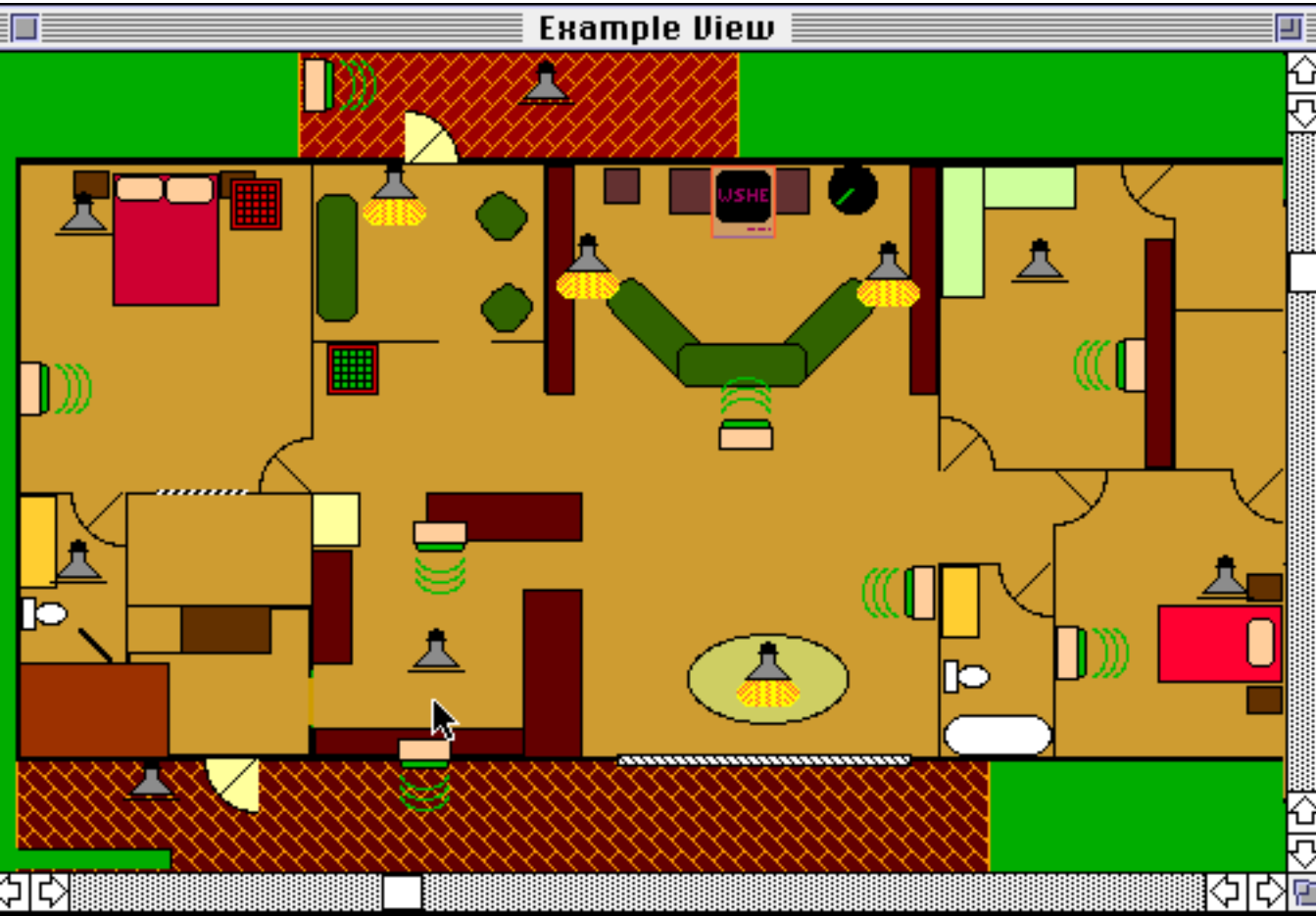
# Graphic Views

The ability to view and control your system by **Graphic Views** and **Icons** has always been one of the most delightful features of XTension.

In the Classic edition of XTension, you could only use images that were saved in the "PICT" or Macintosh icon format. But in the OS X version, you can use virtually any graphic format that the Macintosh can read.

The idea is that you can place **icons** of the units in your system onto a **background**, and they become **hot**, in that they always show the status of the units, and can be clicked to control same.

Below is a simple image that was used in the Classic edition :



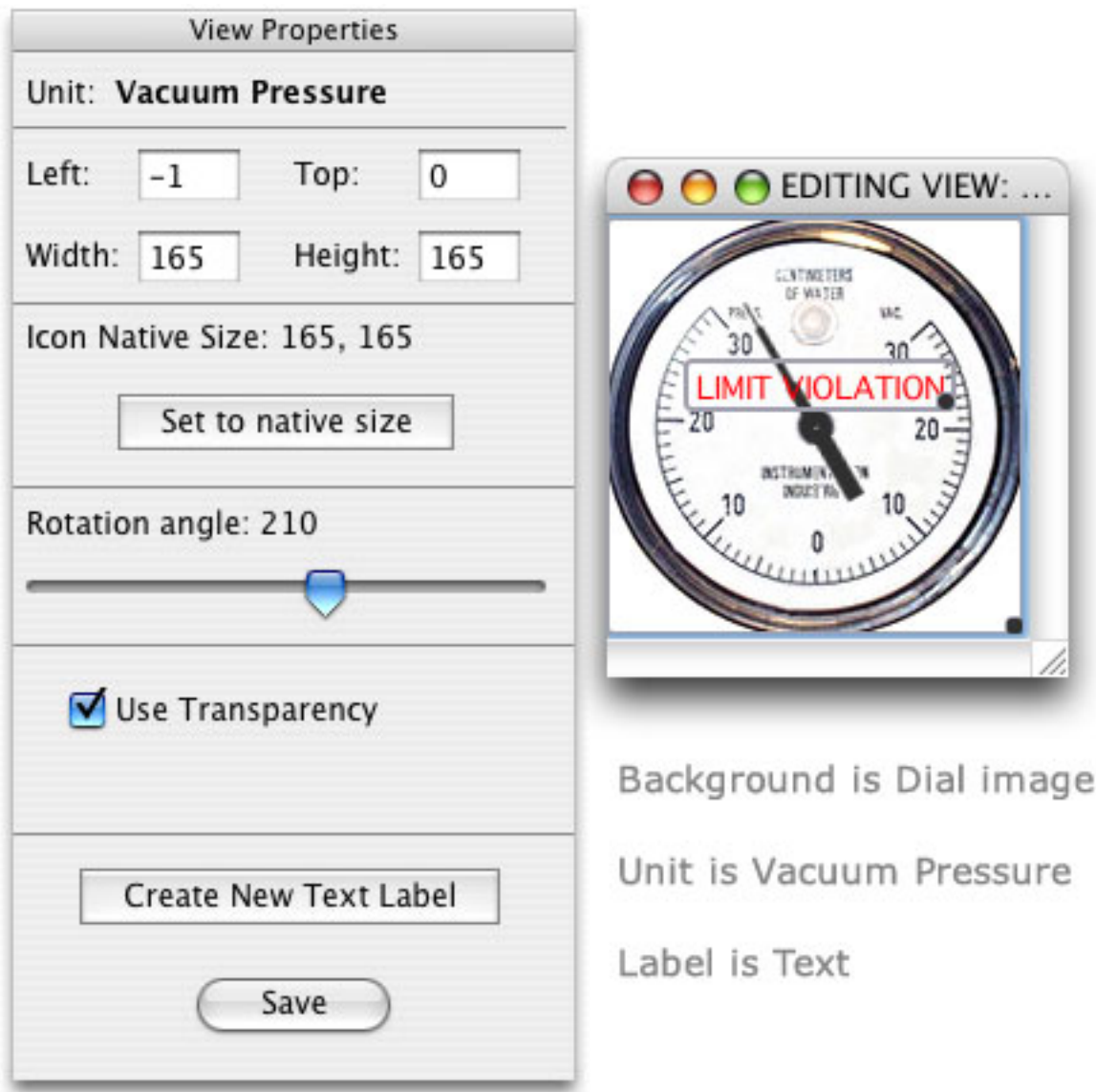
Such a view is exactly replicable in the OS X version, however it is now even more exciting to be able to incorporate all types of background images including **LIVE VIDEO** sources.





From photographs to simple control panels, you can have as many different Views as you need. And you can change the background picture as well as the icons and labels dynamically with scripts !

After the image is created, you may then import it into XTension by pulling down the File menu and selecting '**New View**'. After giving it a name, you can then Edit the View to select any background image or video source, as well as labels and positions and rotation angles etc !!



Background is Dial image  
Unit is Vacuum Pressure  
Label is Text

Once the View is created, you may place icons on it by dragging unit names from any List window onto the View window.

The **current state** of a unit in a View is reflected by its icon. You choose the Icons for any Unit in the Edit Unit dialog. (see that chapter)

You can change the state of a unit from the View window by double-clicking on its icon. And if you choose the 'single click' option, you can just click on it once to toggle it.

You can click and drag any icon around in the View window, to position it. You can also use the Arrow keys to nudge an icon by a pixel at a time.

Any time that you wish to remove an icon from a View, simply select it and press Delete.

Notice that there are several intrinsic XTension verbs that manipulate the display of Views, and even get the results of a current View state : ( for now you should use Script Editor to view the dictionary)

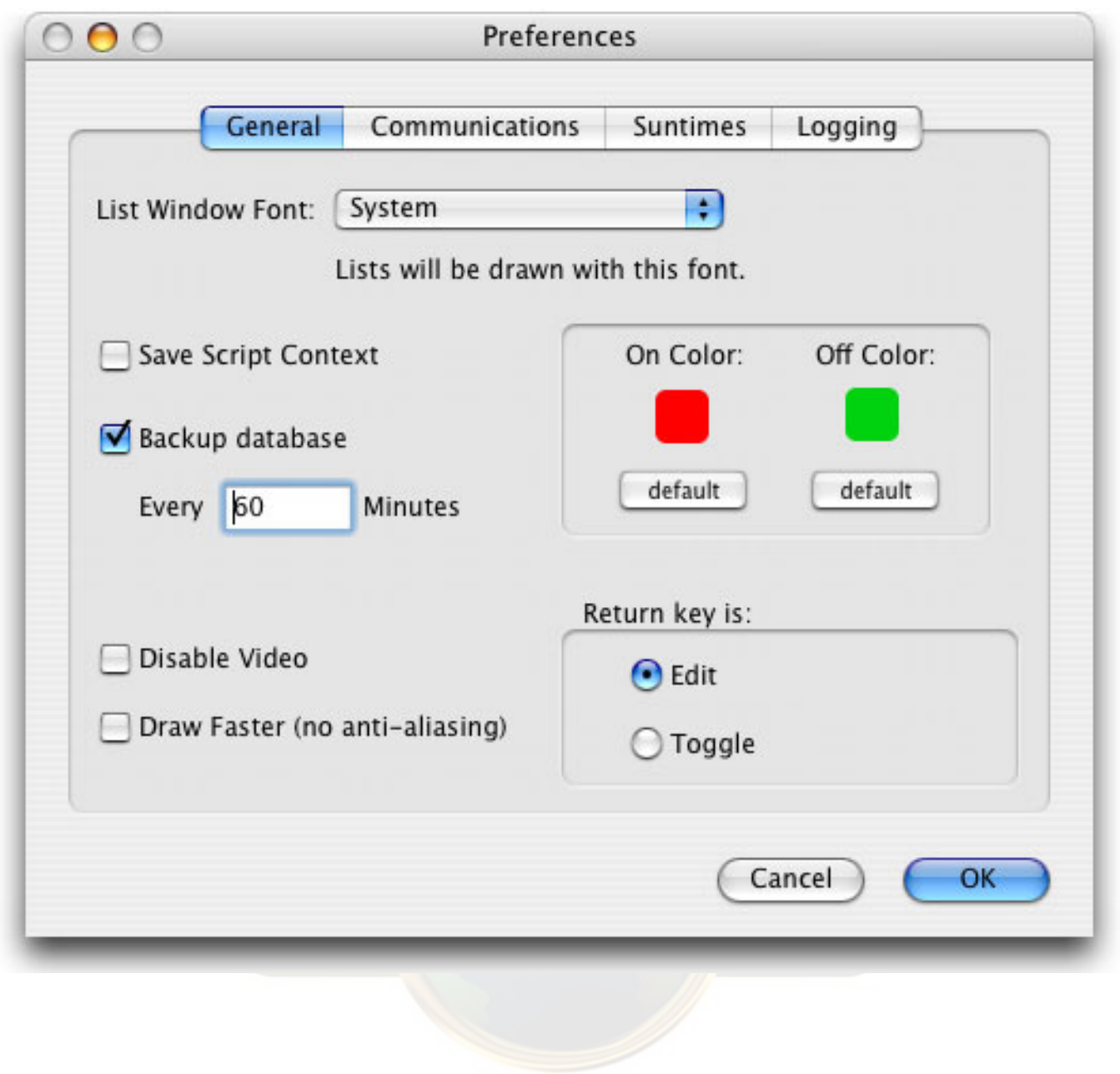
**set label params of**

**put picture**  
**put video**  
**update video for view**  
**release video stream**  
**set on icon for**  
**set off icon for**  
**place icon for**  
**remove icon for**  
**rotate icon for**  
**write view**  
**get raw view data**  
**is view click**  
**get view click coordinates**



# The General Preferences Dialog

This is where you may configure some of the more 'global' functions of XTension. You get here from the Preferences item in the Edit Menu. Notice that there are three 'folders'....



### List Window Font:

Choose the desired font from this popup. It will apply to every LIST window.

### On/OFF Colors :

Clicking on these icons will bring up a selection dialog where you can choose the 'default' color that is used for every unit in 'default' mode, for their ON or OFF status. The Default buttons will choose the colors shown.

### Backup database every XX minutes.

You may choose the period between automatic backups of the database. Although XTension does a good job of keeping up with this, you may want to choose a deliberate period for this.

### Save script context

This Preferences option allows you to specify whether or not you need to save them after they execute, or whenever XTension quits.

If you use 'local variables' with your scripts, saving them will preverve their current values. If you don't use local variables, you don't need it. Leaving this un-checked will improve response and execution time.

Note that if you choose to use this option, every time a script is executed, it will be re-saved to the database. Also, if there



is a 'text' copy of the script in the XTension "Scripts" folder, that text copy will also be re-written to disk.

Exception: If the script exists only as a 'text' copy, ie: it is not in the database, then it will NOT be saved to disk.

If you don't know what this is, then don't worry about it.

**Disable Video:**

This option will simply disallow any LIVE video options throughout the application.

This is to allow for Mac platforms where Live Video is a processing problem.

**Draw Faster (no anti-aliasing):**

Choose this option if you are on a very slow older Mac plataform. The difference in performance can be astonishing...

**Return key is Edit or Toggle in Lists**

With versions 1.7 and later, you can choose how you want to use the Return key when you have selected a unit.

Choosing Edit says that you want to edit the characteristics of the selected unit. Choosing Toggle means that you want the Return key to turn on or off the selected unit, according to its current state.

**Communications Prefs : next.....**

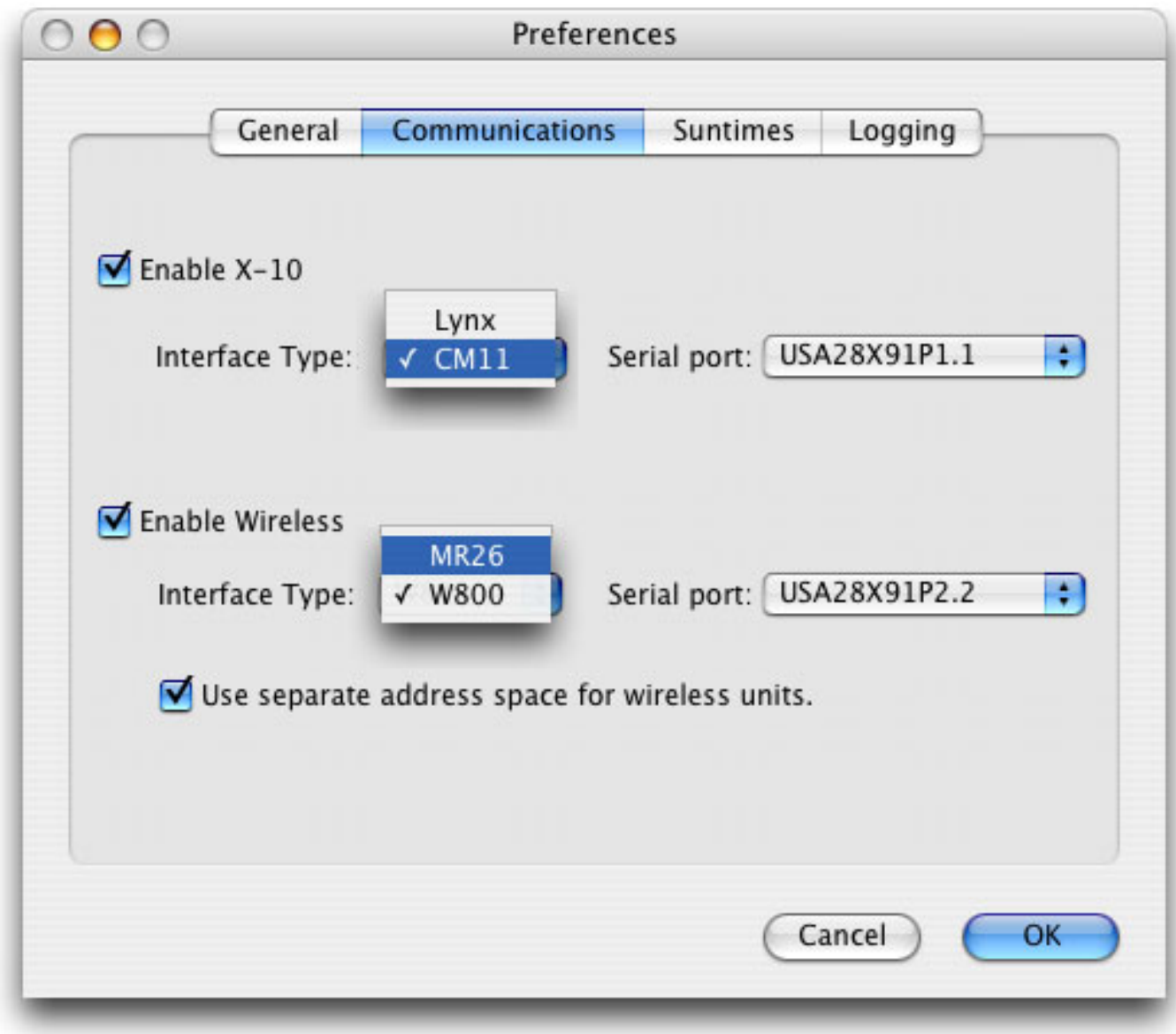


# Communications Preferences Dialog

This dialog allows you to enable/disable the different interfaces, and choose their ports.

In the Classic version of XTension, it was possible to select the baud rate etc, but this is not necessary with the OS X versions.

Note also that with a single version of XTension, you can select either the LynX or the CM11 for the X10 powerline interface, and the MR26 or the W800 as the 'wireless' interface.



Scripting [verbs](#) are provided which do these things dynamically.

## "Use separate address space for wireless units."

Notice this new option. This allows you to specify that all of the 256 X10 addresses can be duplicated according to whether they are 'powerline' based, or 'wireless'.

In other words, you can now have a powerline (plugged in) module on address A1, as well as a Wireless (RF) module also on address A1 (or any of the 256 addresses).

Although not everyone needs 512 addresses, this can make it much easier for those who need that extra address space for things like multiple HVAC controllers (TX15) or other devices that require one or more whole 'house codes'.

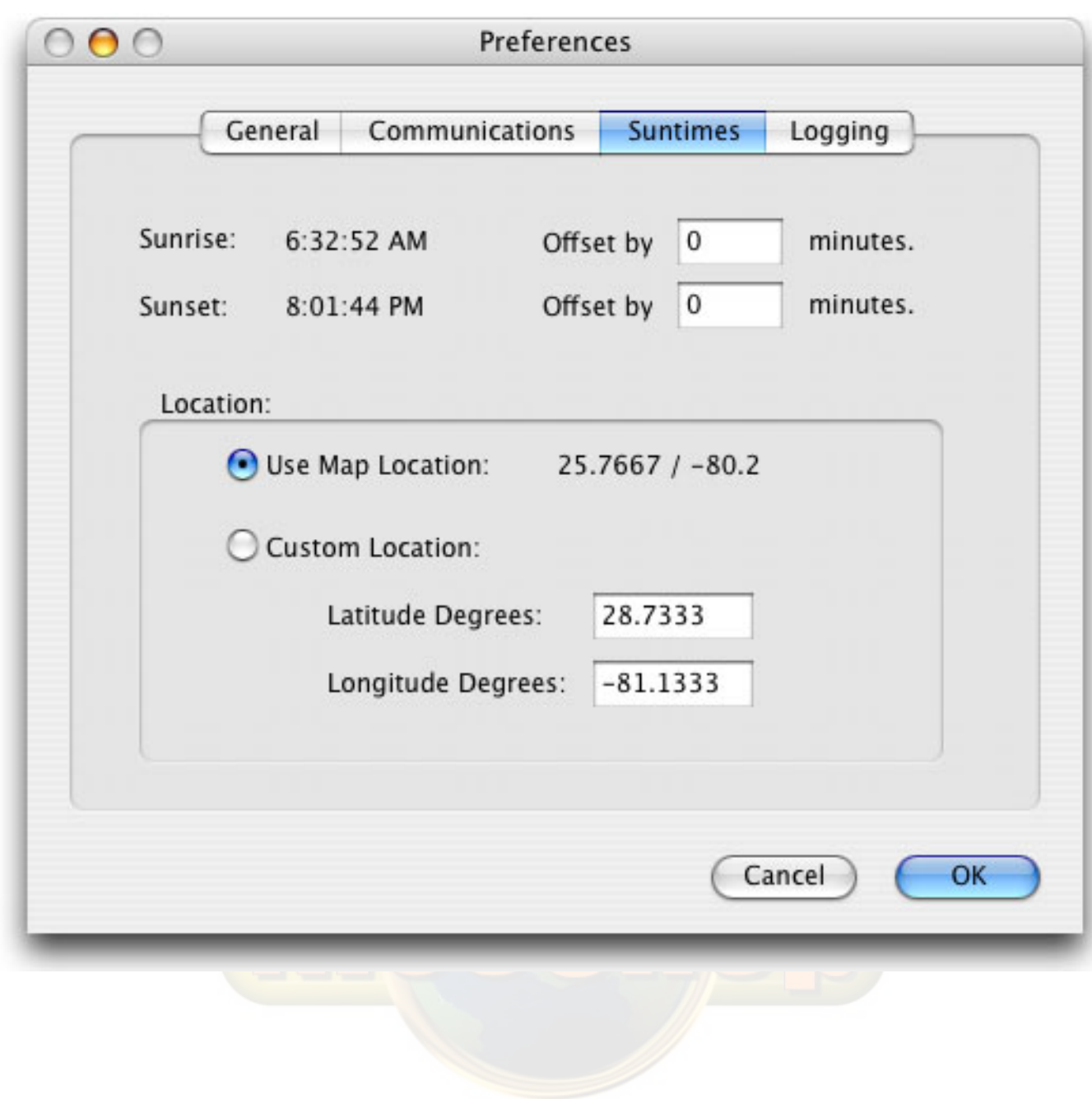
Notice also that this is not affected by the fact that ALL of the 'security module' address space is intrinsically separate from both the powerline and the 'wireless' address space.

**In effect, if all options are enabled, it is possible to have 768 REAL devices in your system !!**



# Suntimes Preferences Dialog

This dialog allows you to configure the sunrise/sunset times for your location.

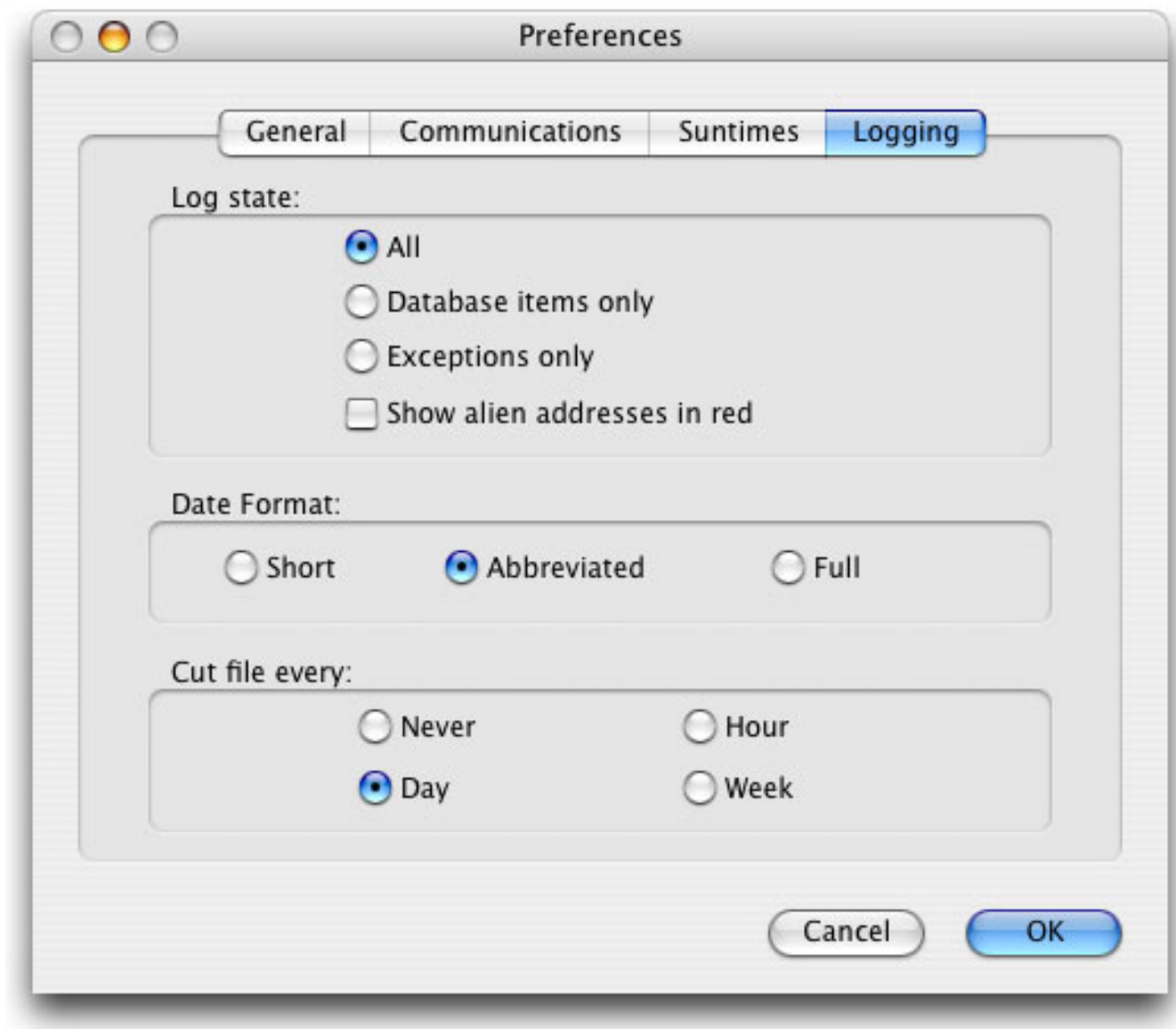


Notice that you can choose either to use the SYSTEM provided Latitude/Longitude according to your Location in the Date/Time control panel, or you can specify a custom (and more precise) location so that your suntimes will be calculated very accurately. (usually within one minute of 'US NAVY')

The OFFSET values are expresses in minutes, and are used only to 'displace' the real sunrise or sunset for situations where you are living in a valley, or a forest etc, and the 'natural light' of sunrise is delayed by some physical barrier etc.

## The Logging Preferences Dialog

Here is where you can change the way that the Active Log Window appears :



Log state controls only which events are displayed in the active **Log Window**. All events are recorded to the Log File regardless. (of course the exception is when the disk has no space)

You can specify how long a period you want to keep in a single log file. I suggest you choose '**daily**'.

Logs are kept automatically by XTension in a '**Logs**' folder in the XTension folder.

You can choose what **level of detail** you wish to see on the the active '**Log Window**', however, XTension always logs all activity to the disk copy of the Log.

By choosing '**Exceptions only**', you will see only **bad** errors **and** messages which are **written** explicitly by your scripts.

By choosing '**Database items only**', you will see all changes to any unit which is in your database. Any commands which are received which are for addresses which are not in the database, will not be displayed.

The option of '**All**' is useful when you are setting up your system, or when you are suspicious that a neighbor has discovered the X-10 system !

**SEE** the chapter on Human Interface Verbs for the AppleScript verbs for controlling these prefs.



## Menu Controls

---

The following section is a simple description of each of the menu items provided by XTension.

### XTension Menu

#### **Visit SHED home page --**

Calls up **Safari** and directs you to the **XTension** home page.

#### **Download the Manual --**

Calls up **Safari** and downloads the most current manual to the XTension home folder,  
and then calls up **Preview** to display it.

#### **Preferences -- command - ;**

Calls up Safari and directs you to the XTension home page.

### File Menu

#### **New Unit -- command-N**

This selection will bring up the Edit Unit dialog from which you may create a new database **UNIT**.

#### **New Group -- command-G**

After prompting you for a new group name, XTension will bring up the Edit Group dialog where you may select database units to be included in the new **group**.

#### **New Event -- command-E**

Normally you would use the Scheduled Events window to create a new event, but this menu item and command key provide alternate ways of getting there.

#### **New List -- command-L**

This selection will ask you for a name and then give you the Edit List dialog where you can select which database items you want in the new list.

#### **New View --**

To create a new graphic **View**, you will be given a Edit View dialog where you select the graphic image file that you want as the background, labels, etc.

#### **Close -- command-W**

This will simply close any window which is 'front'. Note that this does not permanently remove the window, as it may be brought back at any time via the Windows menu.

#### **Print Database --**

This feature will allow you to print or create a PDF document of the units and parameters in the database. This is not complete as of version 5.2

#### **Cut Log File**

This choice gives you manual control of when the XTension log file is archived. This will cause the XTension log file to be written to a folder called 'Logs'. It will be given a name which corresponds to the earliest time stamp recorded in that log. All log files can be read by any text editor.

Note that you may choose preferences which control how often XTension will automatically perform this task.

(You can also do this with an XTension scripting verb)

### **Monitor Only -- command-M**

This choice gives you manual control of the Monitor Mode, as described in the Multiple Macs Chapter. This mode is very important, and the current state of it is always displayed at the top of the Log Window.

### **Import from Classic XTension --**

Initiates a series of dialogs that leads you thru the export/import process when you first migrate your system from Classic to OS X.

### **Look for new Icons --**

Just a helper. If you create new Icons and Views while XTension is running, this is a way to tell it that it needs to look for the new ones and add them to the current selection lists.

## **Edit Menu**

**Cut**

**Copy**

**Paste**

These choices are standard Macintosh, and as you would expect, they are sometimes 'grayed-out' when they don't make sense.

### **Comment selected lines --**

When in the script editor, you can use this option to add (hyphens) to lines, effectively disabling them without taking them out of the text.

### **Edit Unit -- Command - I**

Brings up the Edit Unit dialog for any selected unit.

## **Database Menu**

### **Save Database :**

This selection allows you at any time to **demand** that the database file be saved to disk. This file is normally maintained by XTension according to the period that you select in the Preferences.

### **Archive Database -- not yet enabled**

This allows you to save a copy of your database as an archive. Usually you will use this to save your database after you have created new units or scripts and you have tested them. This file can be copied and renamed at any time to become your active XTension database.

### **Import Database -- not yet enabled**

You can import a 'package' or 'plug-in' system which has been created and exported by someone else. (see chapter on 'Plug-ins')

### **Export Database -- not yet enabled**

Likewise, you can create and export these 'plug-ins' for others.  
( see chapter on 'Plug-ins')

### **Export Database as Text -- not yet enabled**

This feature provides a way of creating a Tab-delimited text file which can be imported into any spreadsheet or text editor. There you can sort it as you wish and use it perhaps as a hard copy map of your devices.

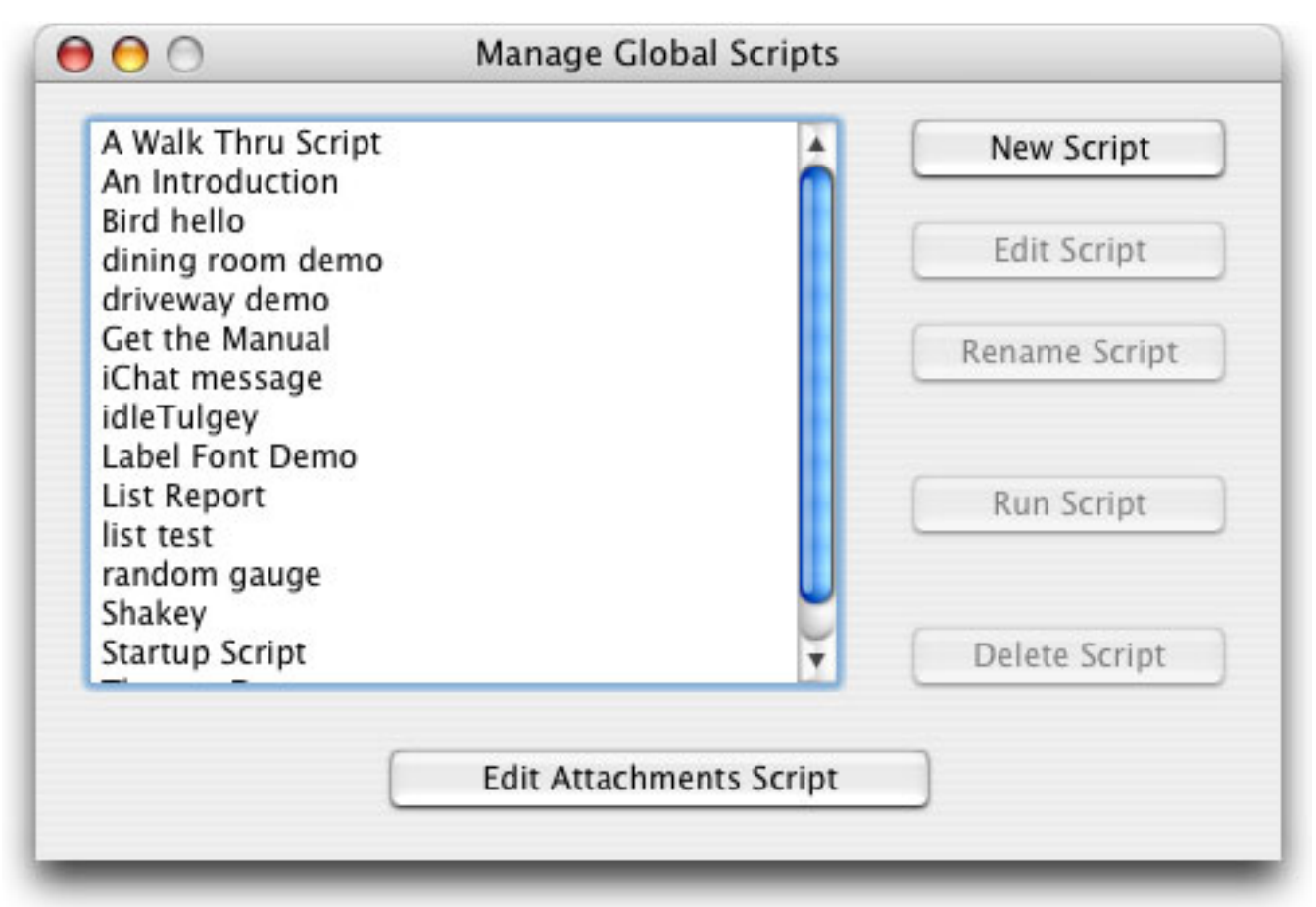
### **Save/Load Snapshot -- not yet enabled**

From the File menu or from a script, you can cause XTension to save the current state of all database units. This file can be Loaded on demand or by script. It is used to synchronize two copies of the database or to review the conditions present at some 'interesting' time. ( see chapter on 'Snapshots' )

Scripts Menu

Manage Global Scripts

These items and their obvious function allow you to create and maintain scripts which are not specifically assigned to the on or off script of a database unit. See the section on scripts later.



Edit Attachments

Brings up a script editor window directly for the Attachments Script.

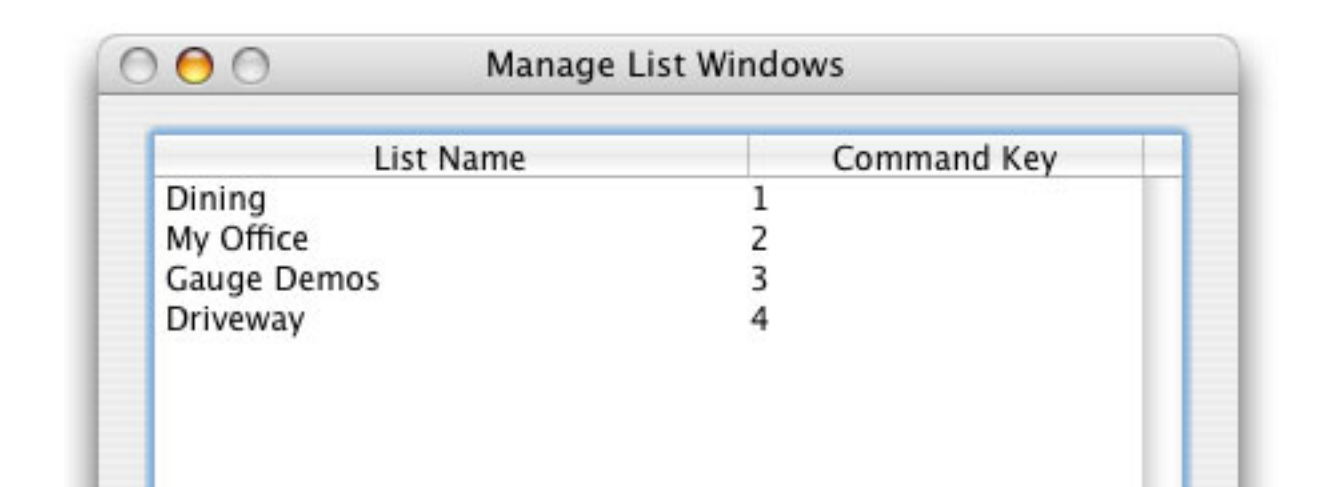
(List of Global Script names)

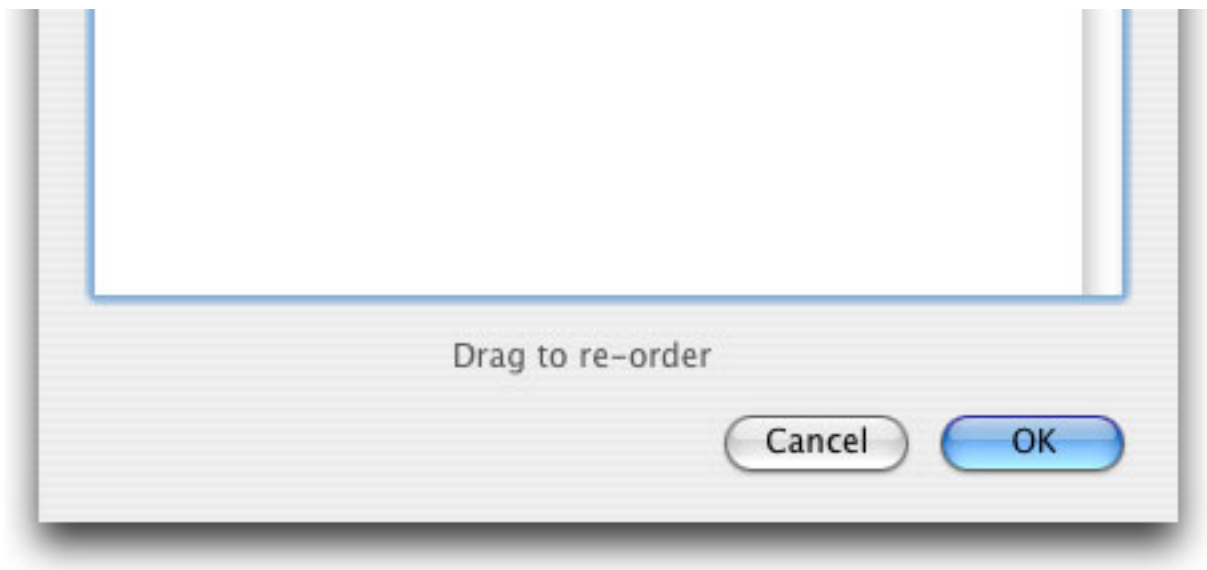
This is a selection list of all of the global scripts that you have created. Simply by selecting one you cause XTension to execute it. Selecting it with the **Option** key down will bring up the Script Editor with that script.

( See the chapter on Special Scripts for information on using AppleScript "Attachments" . )

List Menu

This menu only appears when the front window is one of the Lists that you have created. Here you can create new lists, maintain others, and sort the items in any list by address or by name.





### **Rename List**

### **Delete List**

These two provide the only way to change the name of or permanently remove a List that you have created. You cannot permanently remove the Master List.

## **Windows Menu**

This menu provides a way of bringing any of the XTension windows to the front. The window may be currently 'sent away', or may be just hidden behind another window.

### **Control Panel**

### **Master List**

### **Log Window**

### **Scheduled Events**

### **Command Line**

### **Configure Video Sources (Pop out)**

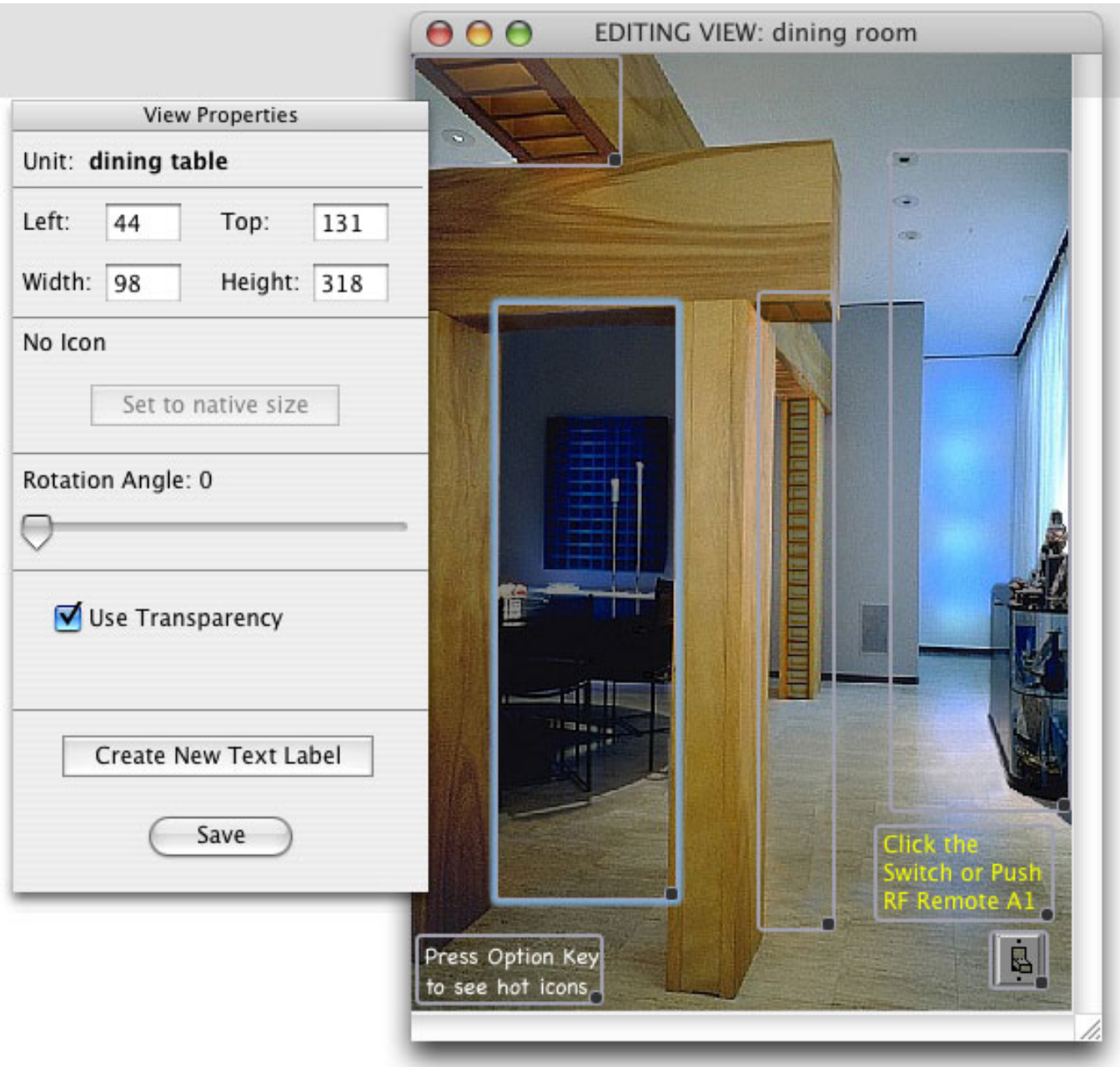
This menu selection lets you release video sources, or look for others.

## View Menu

Here you can edit, rename, delete or bring forward any Graphic Views that you have created.

### **Edit View -- Command - I**

If any Graphic View is FRONT, this choice will bring up the edit dialog for the View.



Here you can drag around Unit Icons, reshape them, choose background transparency, etc.

Note that you can TAB between Items, or between Fields.

When you are finished, just click SAVE or do another Command-I

### **Highlight Icons**

This choice will simply draw the blue rectangles around each of the "hot unit icons" in the View.

### **Rename View**

### **Delete View**

Note that this is the only way to change the name of or permanently delete any View you have created.

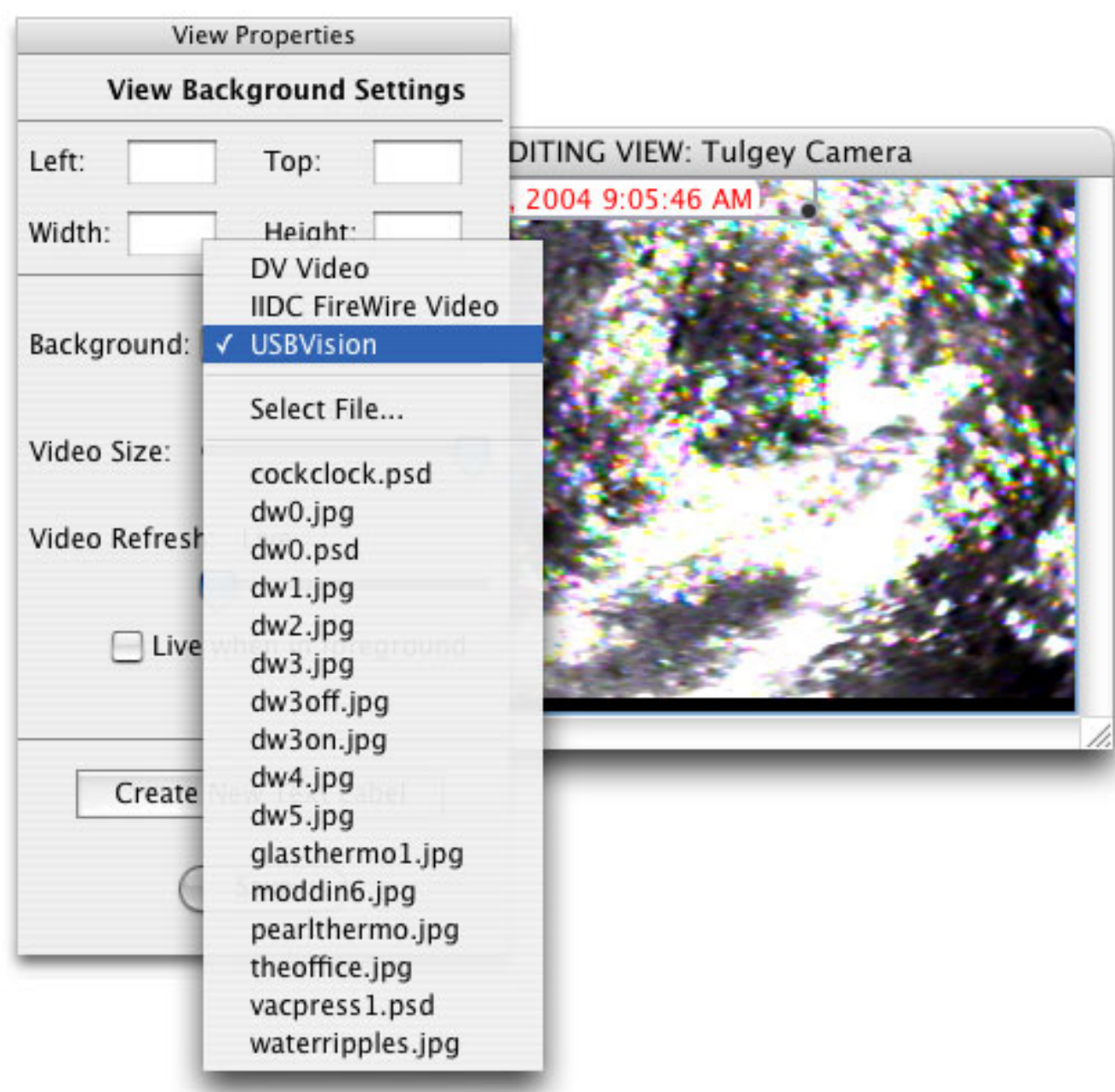
### **New List from View**



This provides a method of collecting all of the units you have placed in a View, and creating a named List which includes those items.

**Edit View**

**Note that you can choose to display LIVE VIDEO as a background to any Graphic View.. !**  
**You can put any live video source into any number of Views, and you can have as many live sources as your Macintosh can support.**



Note that you can change the Size of the Video window, as well as the frame rate, View by View.

## Keyboard Shortcuts and Tricks

Along with the standard menu oriented keyboard shortcuts like cut and paste etc., XTension provide some other useful shortcuts and tricks.

### Apple Human Interface Guidelines:

The up/down arrow keys help you to navigate among lists.

The Return/Enter keys normally 'select' the unit for the most logical effect when in lists and views.

### At Startup of XTension:

- Hold down the "**Option**" key to defeat the **Startup Script**.

This is only necessary when you have created a Startup Script which has an error which prevents proper startup.

### Control Panel:

- Hold down the "**Shift**" key as you click on the 'toggle switch' to issue the same command.

IE: If the selected unit is already ON, a Shift-Click will send another ON to the same unit.

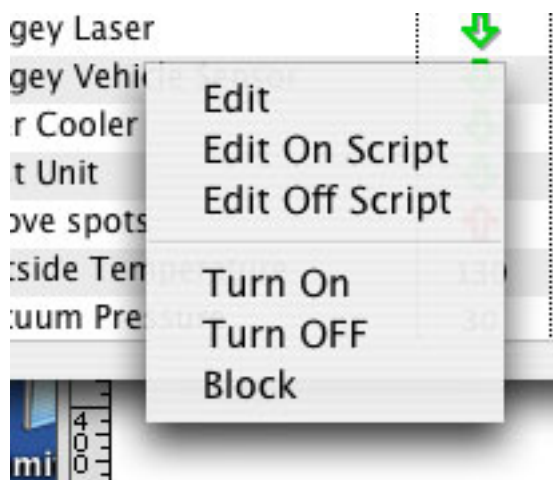
If it's OFF, another OFF will be sent.

### Editing a Unit:

- Select a unit in any view, and hit Return.

### Edit Unit and Scripts:

- Click-Select a unit in any List or View.
- Press the "**Command**" key and Click on the Unit in a List...  
A dialog pops up that lets you select



### Global Scripts Menu:

- Hold down the "**Option**" key and select a script to **Edit** the script rather than execute it.

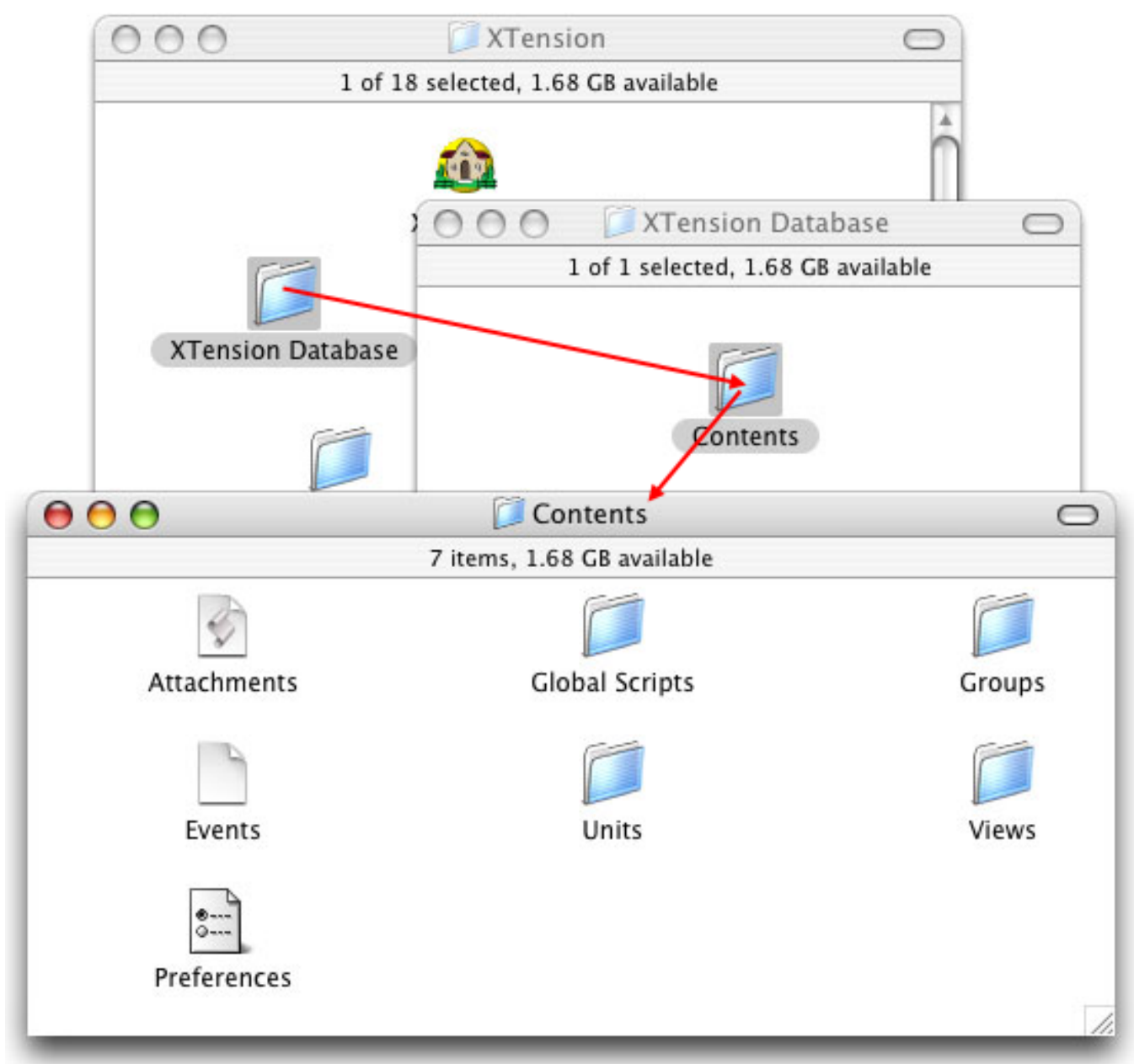
## What's in the XTension database ?

The XTension database contains all of the status and processing information about each of the named **units** you have created for your automation system.

Each of the **scripts** that you create is compiled and saved in a terse form, and is associated with its corresponding **unit** or is a **global** script.

Each of the graphic **Views** that you have imported are saved in a resource within the database file. Other resources in the database file are for **Lists**, and of course the **Scheduled Events**.

What is very nice about the OS X version of XTension is that the database is kept in a very open format, which can be viewed and edited using standard Apple OS X utilities.



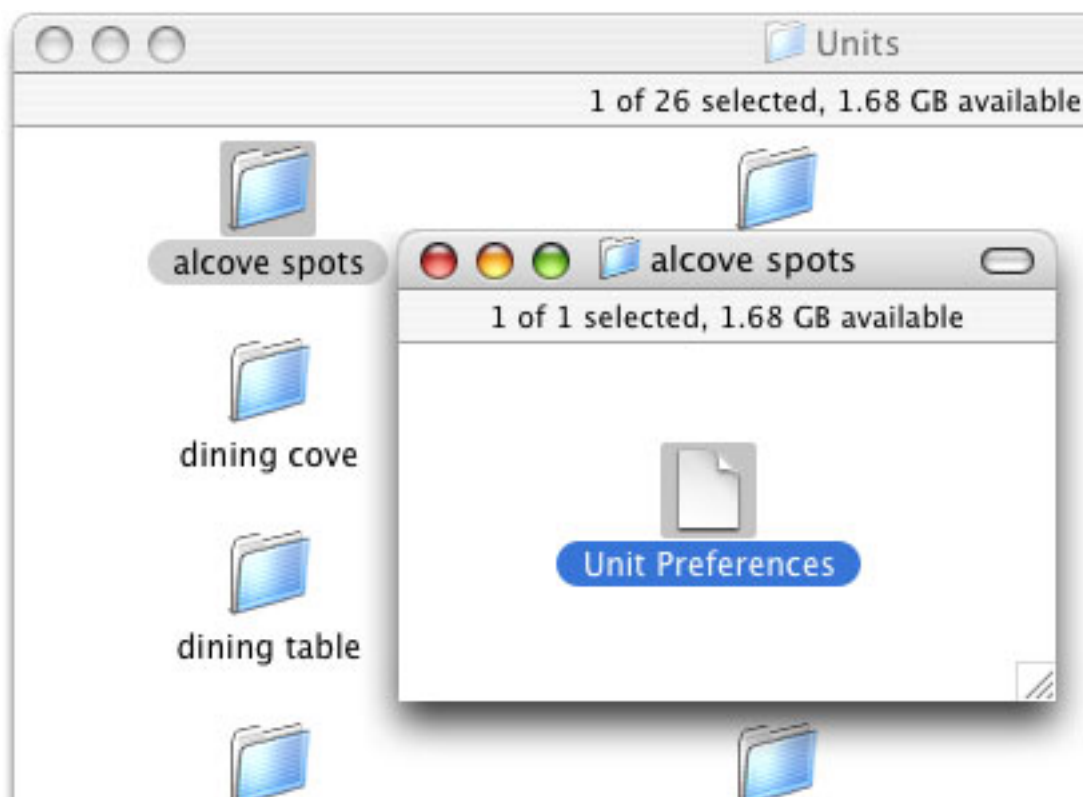
Note: Whenever XTension starts up, it expects to find this file in the same folder as itself.

If it does not, it will create a new empty database .....

This is the only way to create a new empty database.

**Viewing and Editing the Database**

Normal XTension functions are always the preferred editing method, but because the database is kept in XML format, you can view and edit database properties with basic OS X tools.  
... IF you must...



Having opened the database and found a named unit folder, you can simply double-click on the "Unit Preferences" icon to bring up the "Property List Editor".

Property List Editor		
Unit Preferences		
New Sibling Delete Dump		
Property List	Class	Value
▼ Root	Dictionary	23 key/value pairs
Blocked	String	True
Description	String	Recessed lights in the Dining Room
Dimmable	String	False
HardOff	String	False
LastActivity	String	9/22/2004 13:43:45
Name	String	alcove spots
OnIconName	String	alcovespots.jpg
Param1	String	0
Param2	String	0
PassThru	String	False
PresetLevel	String	0
ReceiveOnly	String	False
ReverseLogic	String	False
RFOK	String	False
Security	String	False
Simulated	String	False
SingleClick	String	False
Smart	String	False
Status	String	FullOff
Steps	String	20

UniqueID	String	120093595028.
► UnitParameters	Array	1 ordered object
Value	String	0.

You really want to be careful about changing elements directly here, but given a bit of experience or help, it is at least possible to do this without a lot of bother should the database become corrupted for some reason.





## **Properties of units in the database**

All units in the database have unique **names**. This name is assigned by the user in the Edit Unit dialog when you create or change any unit. Names may be of any reasonable length of characters and symbols, excepting quotation marks. Duplicate names are not allowed.

XTension allows you to enter up to 64K characters of **descriptive text** for each unit. This text is informational only. But obviously, such long fields would have an effect on overall processing.

Each unit has a current **value** or **state** and a **timestamp** for the last time that the unit changed. You can have Analog units (like temperature or humidity) which have a numerical value, and digital units have a state, ON/OFF.

**Units** can have **scripts** assigned to on and off transitions, such that anytime that the unit is changed, a script can be executed automatically. Units do not have to have scripts, and some units may need only one script. Devices such as dimmable lights and thermostats have only a single script. Anytime that an analog device is changed, up/down, on or off, only the ON script will be executed.

All X-10 units of course must have an **address**. This is composed of a single letter between **A** and **P**, known as the '**house code**', and a number between **1** and **16**, known as the '**unit code**'.

Not all units in the database need to have addresses, see **Pseudos** and **Groups** below.

Each unit is assigned a set of **icons** for representation in graphic Views. See the section below on creating and assigning icons to units.

The '**dimnable**' option specifies that this is an **analog** unit and therefore it has a **value** rather than a **state**. This distinction is reflected in the syntax of scripting verbs, and by the way each unit appears in the Lists. If this option is not checked for a unit, then it can only be ON or OFF.

There are also various processing flags for each unit. At times it may be necessary to block certain processes of any unit, therefore each unit has a '**blocked**' flag which appears as an icon in the Lists as well as the control panel.

Certain types of sensors have what is termed 'reverse logic'. In other words, a device sends an OFF state to signify an alarm, and ON means that the sensor is relaxed. If you check the '**reverse logic**' box in the Edit Unit dialog, then you can write scripts for this device which use 'positive' logic, where an ON is an ON and vice versa.

The '**receive only**' flag provides a method of preventing any script from changing a particular unit. Thus it can only be changed by receipt of a command from an X-10 device such as a wireless remote.

## **Types of units**

### **Commands and Measurements**

**Commands** are thought of as actions which turn things on or off, or set a lighting level. **Measurements** are actions which convey an event such as movement sensed, or temperature, or daylight/darkness.

Although it is possible to issue any address from any wireless or night stand

controller; in the design of a good system, it makes sense to make a distinction between **commands** which are issued by scripts or ad hoc from remote controls, and **measurements** which are the output of sensors.

Scripts written to serve Measurements are “**Reactive**” , and those which serve Commands are called “**Prerequisite**”.

( We **react** to a motion sensor, or a temperature,  
but we impose **prerequisites** before turning on the electric fence. )

## Groups

From the New Group dialog, up to 50 of the named items in the database may be selected at will and given a name which may then be used in scripts to control all of the units in the group collectively. Up to 100 such groups may be created.

Although groups are generally made up of units which have X-10 addresses, the group itself has no hardware address.

You may make groups which include units which are also included in other groups such as :

**All Front Lights** -- includes all front lights  
**All Rear Lights** -- includes all rear lights  
**All Outside Lights** -- All front and rear lights

You can create a non-sensical group, one which may pass the script compiler, but may not do what you want :

Group name: All Walk Lights  
Members of group:  
RearWalkLights (ok to choose a group )  
FrontWalkLight ( ok choice )  
LRlamp#1 ( not a walk light )  
FrontDoorMotion ( this is a sensor )

A script which says : Turnoff “All Walk Lights “ would turn off each of these items in the database, and issue commands to them, but FrontDoorMotion doesn't make sense here.

The **time stamp** and **value** of any **group** is taken from the last time that the **group** was commanded. All members of the group are changed and time stamped individually at that time.

If any member of the group is explicitly changed thereafter, it does not cause a change in the status of the group.

It makes sense to create groups of similar functions. A group of living room lamps which are all on DIM-able modules, can all be set to 50% by issuing a script:

**DIM “All Living Room Lamps” to 50**

**Pseudos ?** ( what else would you call 'em ? )

Occasionally you need to be able to set flags which indicate modes or conditions, counters and values. Concepts such as “**Daylight**” or “**Normal Working Hours**” or “**Monthly Rainfall**” are really complex or calculated functions.

They may not be related to a particular X-10 device, but may be the result of multiple conditions or changes of sensors and commands.

"Normal working hours" might be the product of several functions. It must be a work-week day, the time must be between beginning and end of the normal work day, it can't be a holiday, etc.

Thus, scripts for sensors can test the **pseudo** "Normal Working Hours" to determine whether to cause an alarm and phone call, or just to ring a chime.

Pseudos can be discrete or analog. For example, there is a rain gauge which sends an X-10 signal for each 1/10 inch of rainfall. The script for that unit would add a count of 1 to the pseudo "Monthly Rainfall" which would be further used in scripts which determine how often to turn on the sprinklers.

The only thing that distinguishes a pseudo from other types of units in the database is that **it has no hardware address**, and groups appear in outline font.

Pseudos can also be members of groups.



## The Log Window and Log File

It depends on how you **choose** to view it !

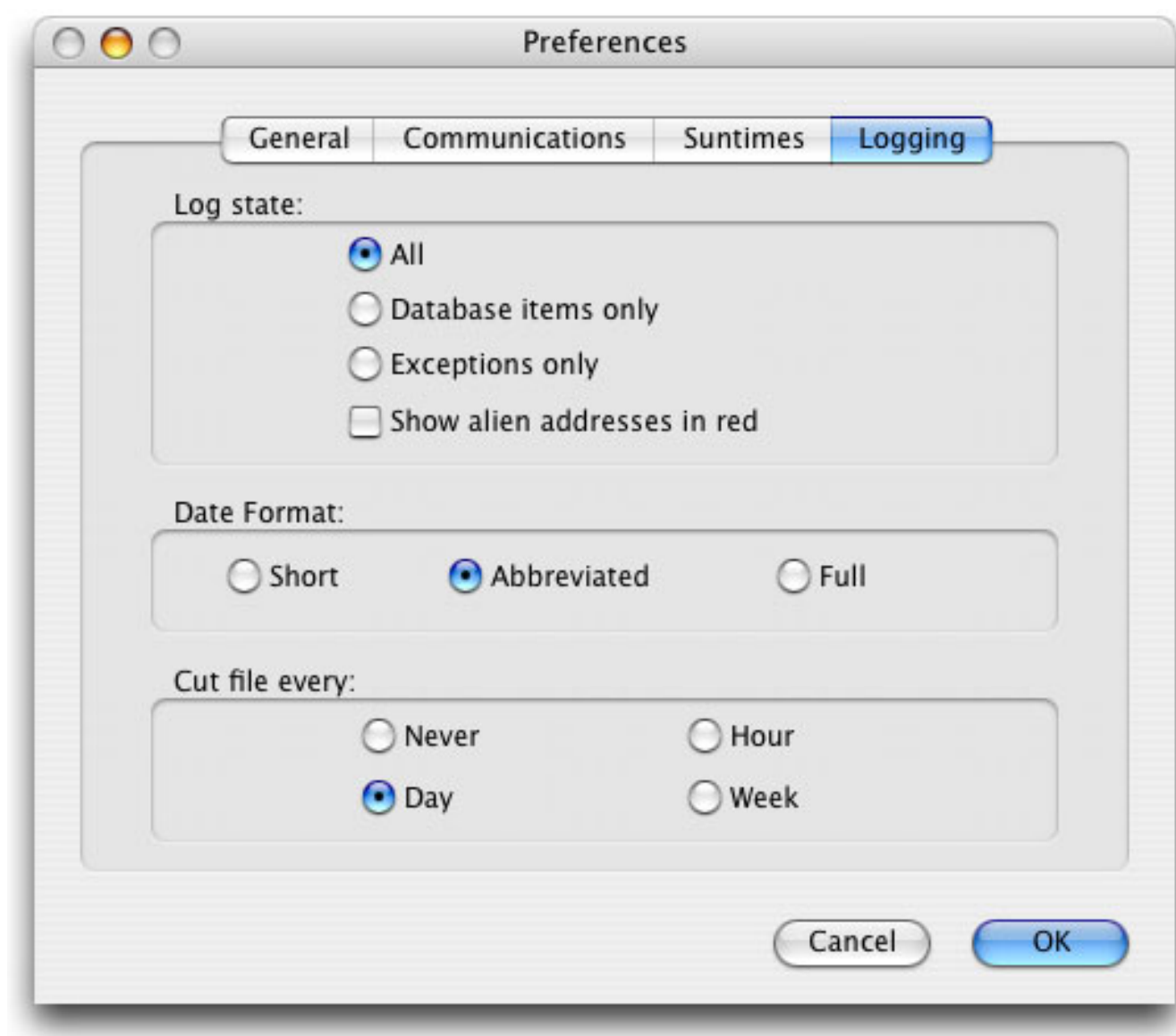
XTension maintains a detailed file system on your **disk** regardless of the level of detail that you choose to display in the Log Window.

The **Log Window may be put away**, but it can be recalled and **is permanent**. The Log File will always be written to your disk, and will be archived in a folder called "**Logs**" in the same folder as XTension.

There are times when you want to see ALL of the activity of the devices and sensors of your system, like when you are testing some new features you have added.

Sometimes, you really don't want to be bothered by a lot of unnecessary messages written to the XTension Log Window.

You may only want to **see** the 'exceptional' events.  
And you may want to see some events in a specific **color** !



XTension normally displays a log window which shows system events in a text format with time stamps. The amount of detail in the window is chosen in the Preferences dialog, and by way of specific '**set logging**' verbs in your scripts.

From the Preferences dialog, **Logging** page, you can choose to view '**All**' events. This will cause XTension to display everything that happens on the X-10 bus or within XTension. This mode is very useful when you are developing your system. Watch for things that are not reasonable.

The '**Database items only**' choice will cause XTension to ignore extraneous events, but will report all events related to the items in your house (database). This mode is useful if you like to see everything that happens, but may be too much detail for most of us most of the time.

Choosing '[Exceptions only](#)' will make XTension display only very bad system errors and explicit messages that you write explicitly to your log with the 'write log' verb (see Human interface verbs)

This is the '[normal](#)' Logging mode that you will use when things are stable in your system.

### **Do I want to make a single Log file forever ?**

You **will** want to choose how often you want to '**cut**' the log. This only a matter of personal preference, but allows you to keep an archive of past logs. When you 'cut' the log, XTension writes the log file to a folder called "Logs" in the XTension folder. The name it is given corresponds to the date/time of the first entry of that log file.

### **Should I be concerned about disk space ?**

Yes, since XTension often runs on that Mac in the closet, and because it's likely that you will ignore it, it is possible that the Log Files will eat up the disk space over a period of time.

The easy solution is of course to just check it out every month or so, but if you are lazy and have another Mac with more disk space on your network, you can create an 'alias' of a folder named "Logs" which is on another Mac (or another volume on the same Mac), and just move that alias into the XTension folder. XTension will see it and save all archived Logs in that folder (whenever the log is 'cut').

### **Please NOTICE:**

If the disk space on the volume where XTension is saving the Log files should ever run out, XTension will write a message to the Log Window, describing the problem, and will then terminate writing to the disk-based Log file.

This will not stop XTension functions, but if you have been ignoring the disk space already, then you might also miss the fact that XTension is no longer logging to disk.

If you discover the problem, and simply restart XTension, disk logging will automatically be restarted.

If you want to take action of your own, you must create a special global script named "**Errors Script**". There, you must test for the disk is full error condition and either compress old log files or delete them in order to free up disk space.

Once you have freed up space, you can then use the verb "set disklogging" to re-enable disk logging without stopping XTension. See the section on "Special Scripts" for more information about the "Errors Script".

### **I'm really bored with 'Blue' :**

There is also an option to the '**write log**' verb which allows you to specify the **color** of the entry in the Log Window. see Human interface verbs.



# Scripting

## Why Scripts ?

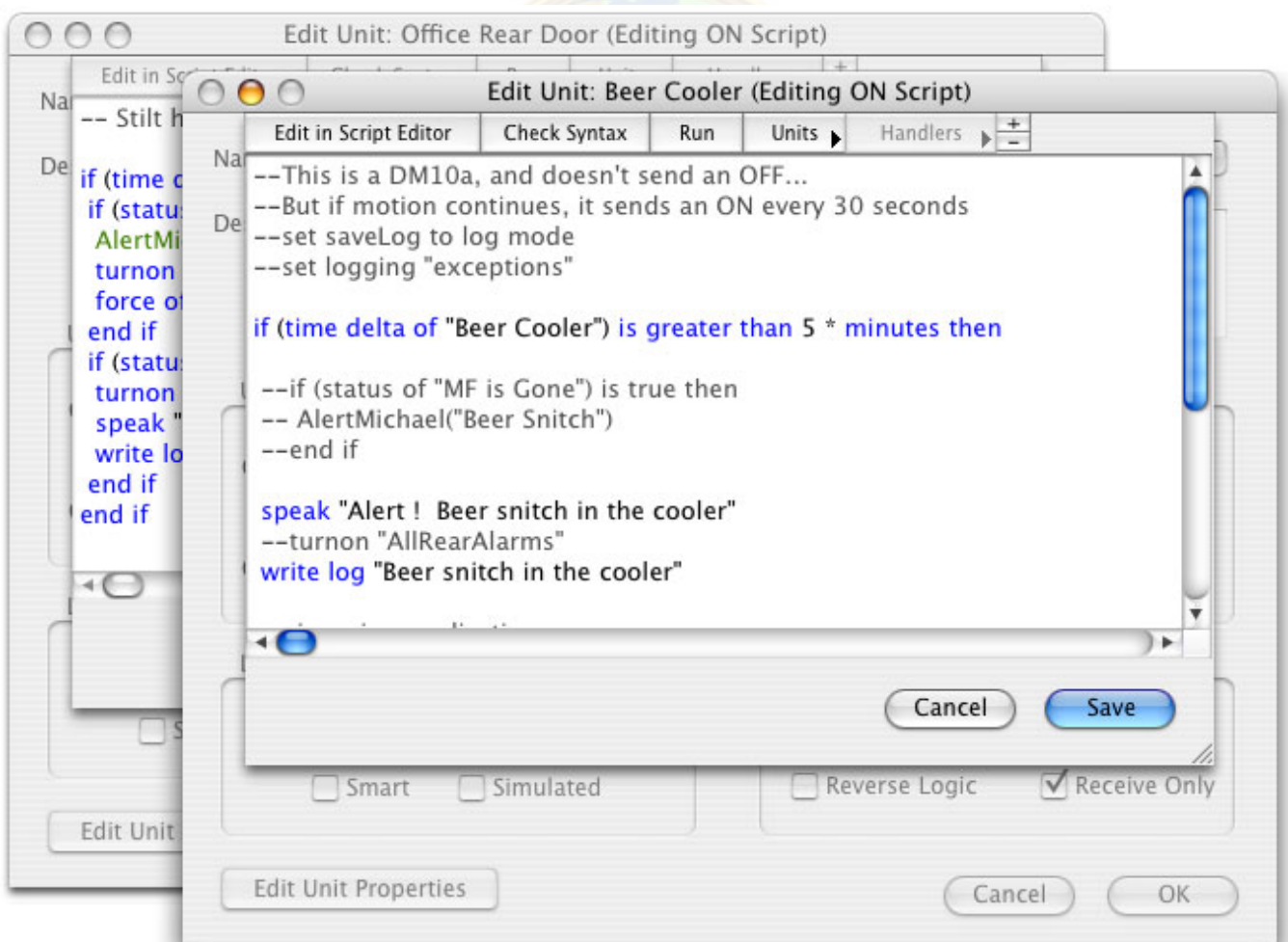
Scripts provide the real flexibility in all good automation systems. XTension scripts are written in Apple's standard scripting language AppleScript, and can be simple or very complex. Although simpler systems may be constructed using a 'rules-based' approach, the rules will always be more restrictive than a natural language approach.

XTension provides several verbs and nouns, and AppleScript provides an extensive dictionary of verbs and constructs.

XTension calls upon the AppleScript compiler to test and execute scripts, and the AppleScript server calls upon XTension to process phrases that it serves. Therefore, you may integrate XTension functions into scripts which span widely different systems and applications.

For example you might have a script for an accounts receivable system which tells XTension to ring a bell each time you add another \$1000 to your total income. Or, gather callerID information in another application, and have XTension put the entry in the log and sound alarms (or not) according to who is calling.

Using XTension and AppleScript, you can personalize a home system or create a complete facility management system for a shopping mall or greenhouse.



## Scripts are really programs.

Ignoring all of the infra-structure beneath them, each script in itself is a tiny

program. Its purpose is either to **react** to some autonomous event such as a movement sensor, or to **guard** any attempted change to some controlled device such as a hot plate.

If a movement sensor indicates an alarm state, we may want to do more than just turn on a light. Devices exist of course which do this automatically, but you still want XTension to notice and change the unit status in the database and perhaps take action.

You may want to do different things according to the time of day or day of week or whether other movement sensors have indicated that there hasn't been movement 'inside' for more than an hour.

### **Global Scripts**

Sometimes it is useful to have common scripts which can be executed on demand.

The **Scripts** menu of XTension allows you to create a **global** script, give it a name, and save it so that it is available in the menu or to any other script on demand.

These scripts are called Global scripts because they can be executed from the menu and from other scripts.

They can even be executed by programs and scripts from outside of XTension !

### **Special Scripts**

You may create two specially named scripts which XTension pays attention to whenever XTension starts up or is told to quit.

If you create a **global** script named '**Startup Script**', XTension will run it everytime that XTension starts up. Likewise, the script named '**Shutdown Script**' will be executed any time that XTension is told to quit. Neither of these scripts is required, however you will find useful applications for them.

See the **Special Scripts chapter** for more of these.

### **AppleScript**

The **MacOS** provides a powerful scripting facility with **AppleScript**. Almost anything that can be done with the Mac or a Mac application can be controlled or enhanced by scripts which can in themselves be 'applets'.

Many common verbs and conditional constructs are provided by **AppleScript**, while applications can publish dictionaries of their own verbs for special functions.

It is even possible for you to create additional 'commands' which are just other scripts ( like subroutines ) !

There are AppleScript editors from Apple and third parties which can record actions and compose scripts to perform some very complex functions.

XTension provides a simple **script editor** which is sufficient for writing scripts for your automation system, thus you do not need to use an external script editor. You may of course write scripts in any script editor, and import them into XTension. Scripts created by XTension may be saved as text and read by any text or AppleScript editor.

There is not enough space in this manual for inclusion of the complete dictionary provided by AppleScript. There are many books and manuals already available which will do a good job of tutoring you in AppleScript. ( get Danny Goodman's AppleScript Handbook )

You do not need to be an AppleScript guru to write scripts for XTension. Most of the scripts you write need only simple if/then/else statements along with verbs and constructs from XTension's dictionary.

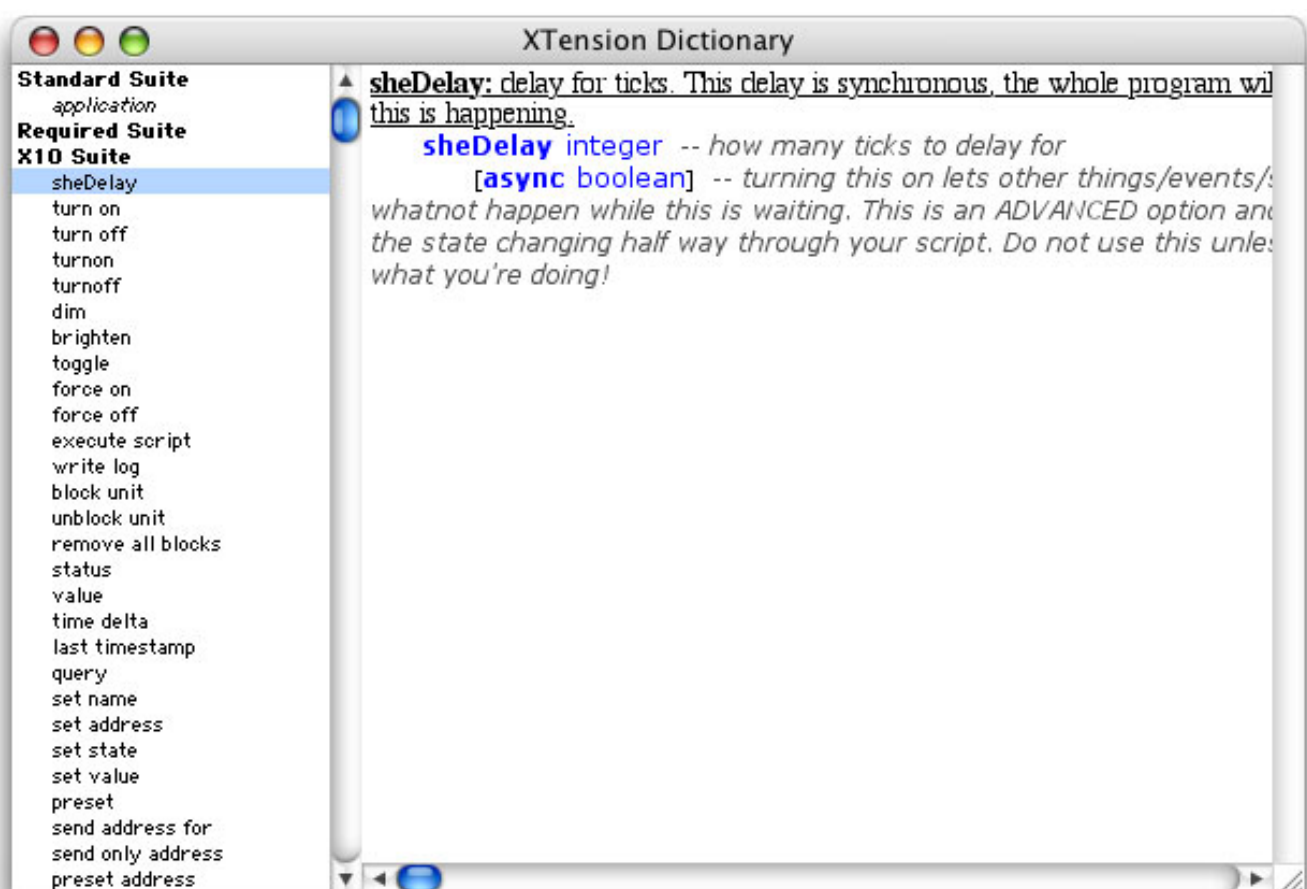
```
if value of "Temperature" is greater than 70
    turnoff "Heater"
else
    turnon "Heater"
end if
( this is a stupid example, just shows simple form )
```

## **The XTension dictionary**

XTension provides verbs and constructs which are specific to process control, and perform reasonable functions on items in the database.

XTension recognizes all of the named units which are in the database, and will support their use in AppleScript statements as well as the verbs in the following chapters which are unique to XTension.

You should know that anytime you want to view the entire dictionary of XTension verbs, just call up Apple's Script Editor and ask it to Open the Dictionary of XTension...

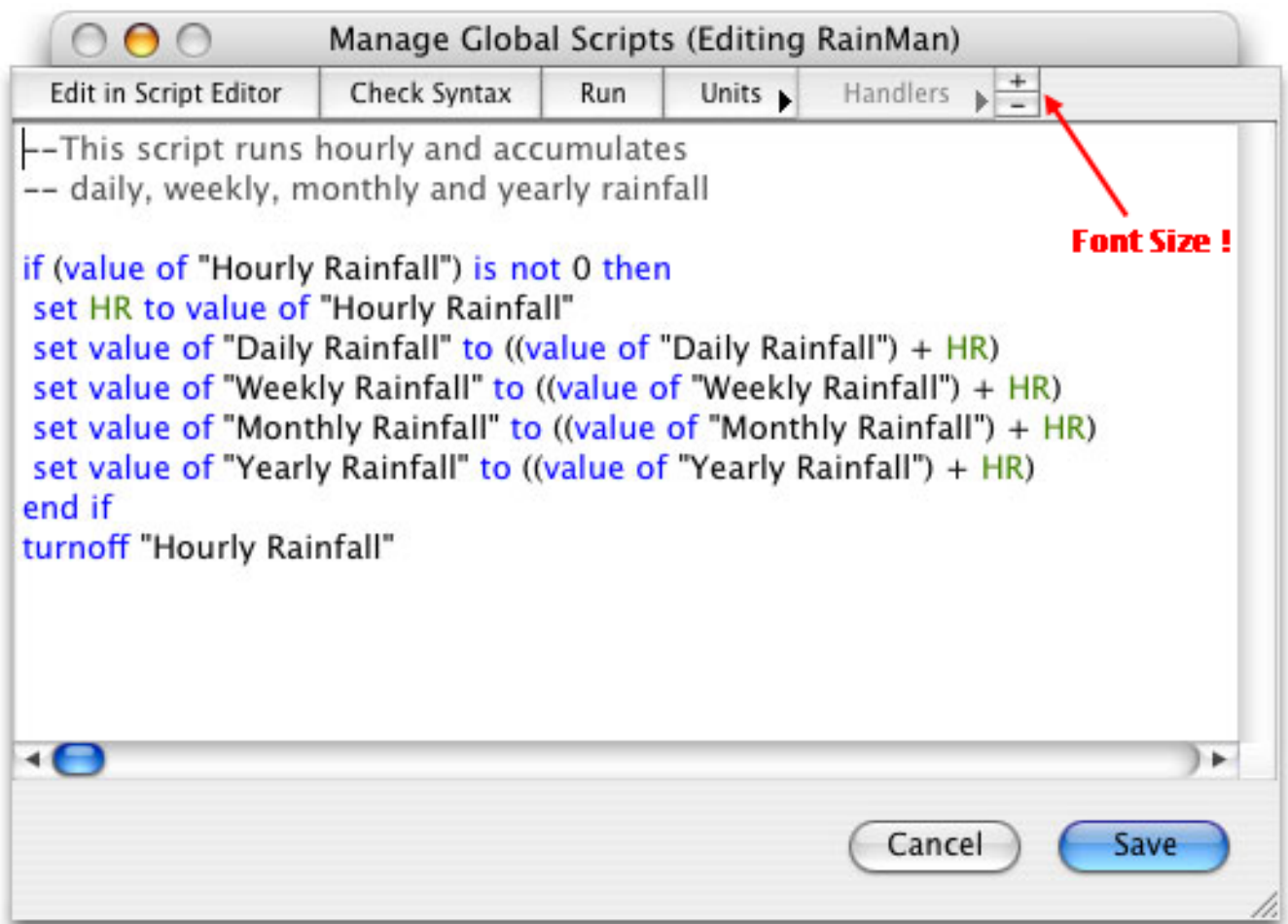


## The Script Editor

XTension is one of the very few applications that provides an intrinsic script editor.

The idea is to remove any excuse you might have NOT to modify your scripts at your slightest whim.

Every little impediment like having to start up another application, open the file, modify, test, and save, is the anathema to keeping your system perfectly tailored to your personal preferences and maintaining that ever so important Spouse Approval Factor !



The first thing you should notice is that there is a button in the upper left corner that lets you escape to Apple's own Script Editor, should you just prefer that program.

Our editor acquires its color-scheme from those preferences that you set in Apple's Script Editor... under the Preferences dialog.

Both editors allow you to use any of the intrinsic Scripting Additions, OSAXen, etc.

Both allow you to 'tell' other applications to do things, check syntax and 'run'. The main Script Editor has more tools for debugging in more complex situations.

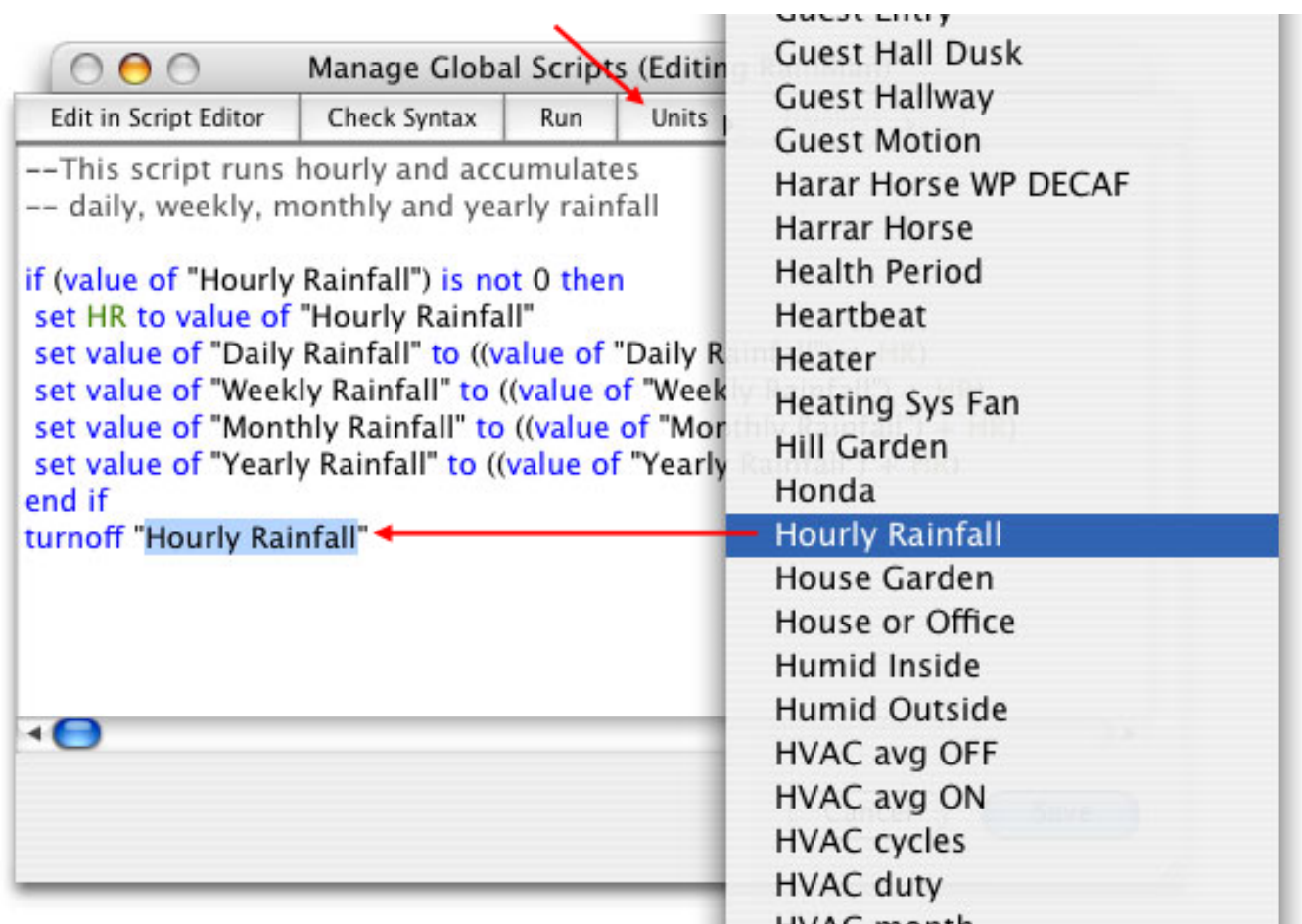
**But there are some things that our editor offers that Script Editor doesn't ...**

The main Script Editor is not aware of the **UNITS** in XTension's database, and whenever you want to use any of XTension's **VERBS**, you must use the 'TELL application' phrase or Script Editor won't know where to look for them. (That is ONLY when you expect to edit and RUN scripts from within Script Editor).

## The Units pop-up



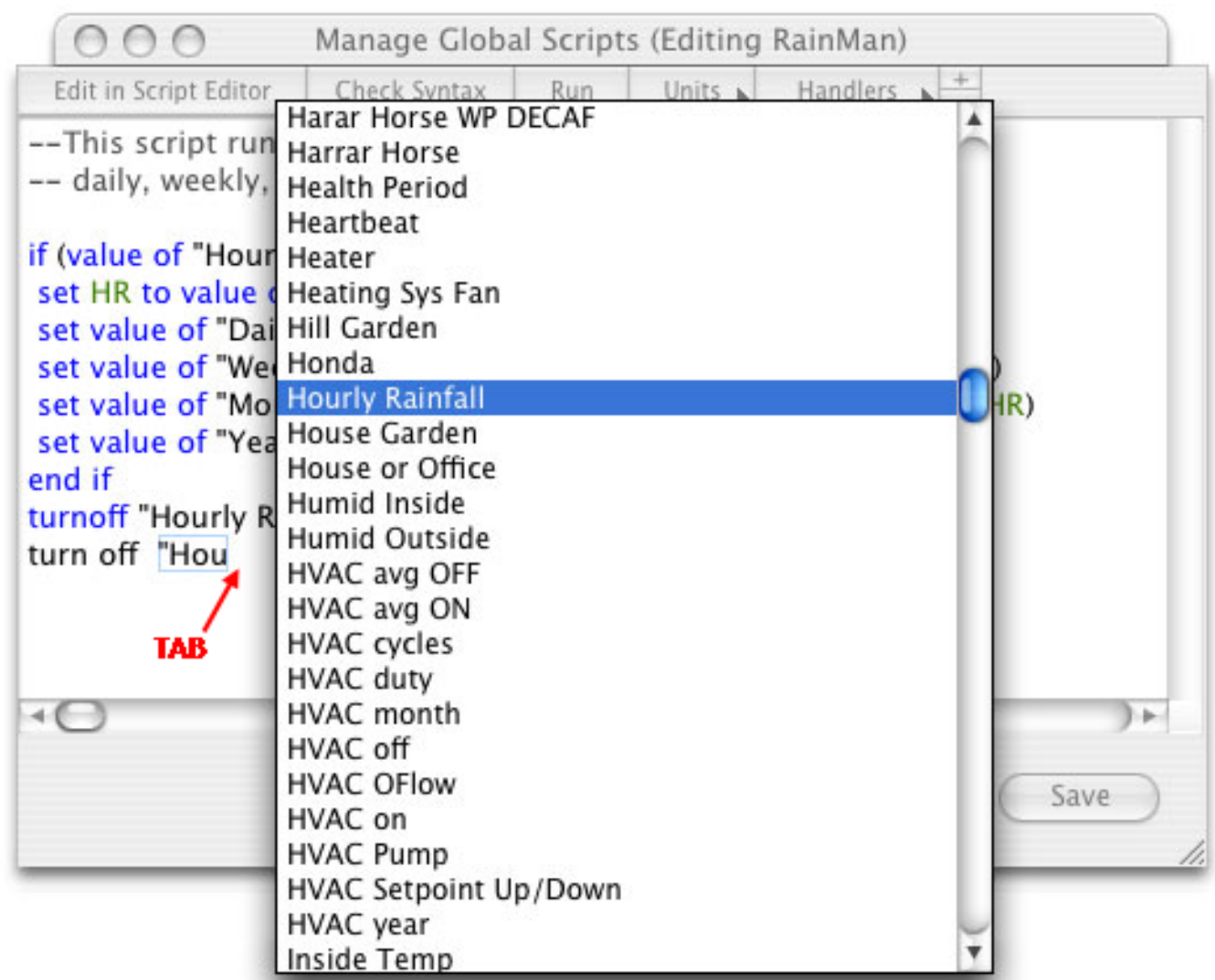
Whenever you are typing a script line that requires the name of a database Unit, and you don't remember just what the name is, click on the 'Units' button in the header above the script area:



Scroll down till you see the Unit name you want. **DOUBLE CLICK** on it, and that name will be placed at the 'cursor' location, along with enclosing quotation marks !

**The Units auto-find**

Likewise, whenever you need a Unit name and can remember the first few letters, just type those and then press the **TAB** key...



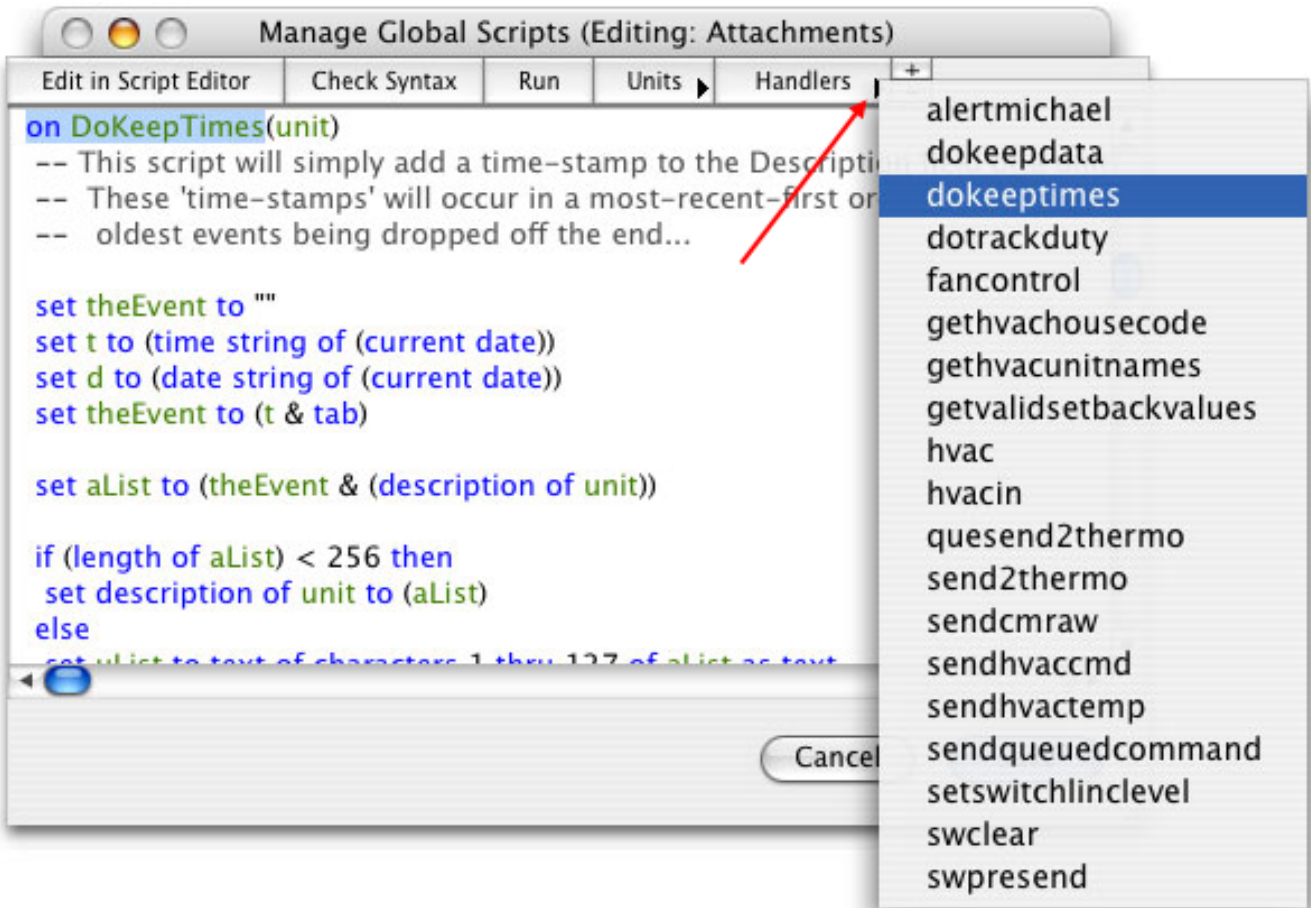


XTension will find the first unit name with matching letters, and again, all you do is double click on the desired one.

### The Handlers pop-down

Another thing that is very nice is the ability to show 'handlers' that are in any script.

"Handlers" are really 'subroutines' or 'verbs' that are written in AppleScript, and may be used in other parts of the script. Or even by other scripts in the case of the **Attachments Script**. (see that chapter)



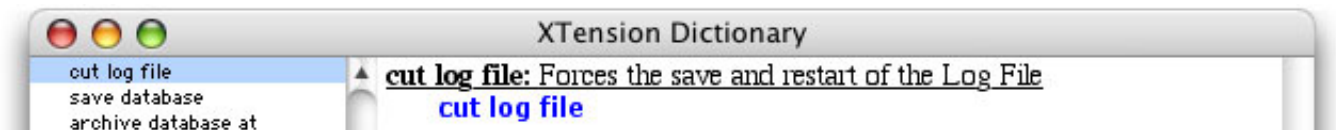
If the script contains any handlers, clicking on the 'carat' will yield the pop-down where you can select the name of the handler. Clicking on the name will cause the script edit window to be scrolled to that handler.

# About the Verbs and 'Nouns' and Syntax

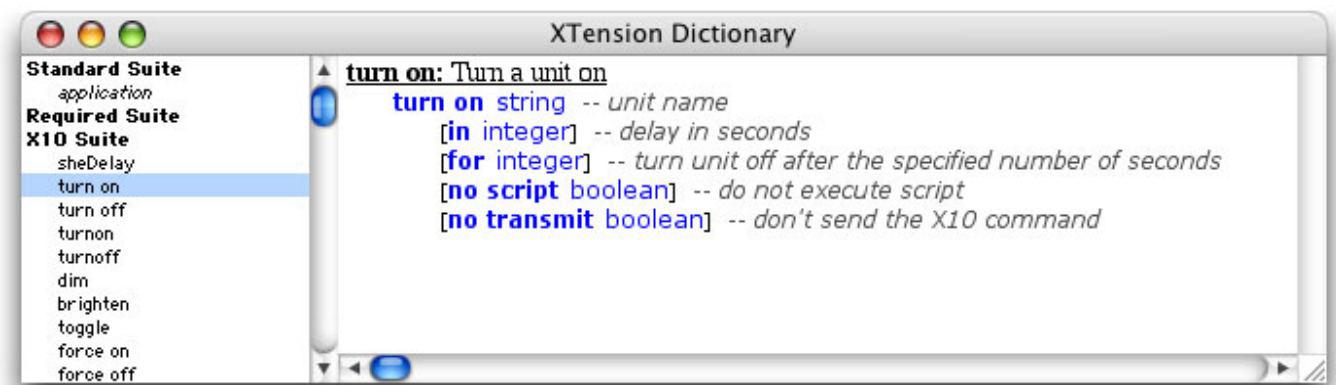
XTension's verb dictionary is the result of many years of careful attention to the needs of its users.

Often you will find that there are verbs that seem to have overlapping functions, or even duplicate functions. This is our way of recognizing that not everyone 'thinks' in the same way, just as everyone has a different vocabulary and style of speaking.

Using the very powerful architecture of AppleScript, we have created verbs that are simple and direct:

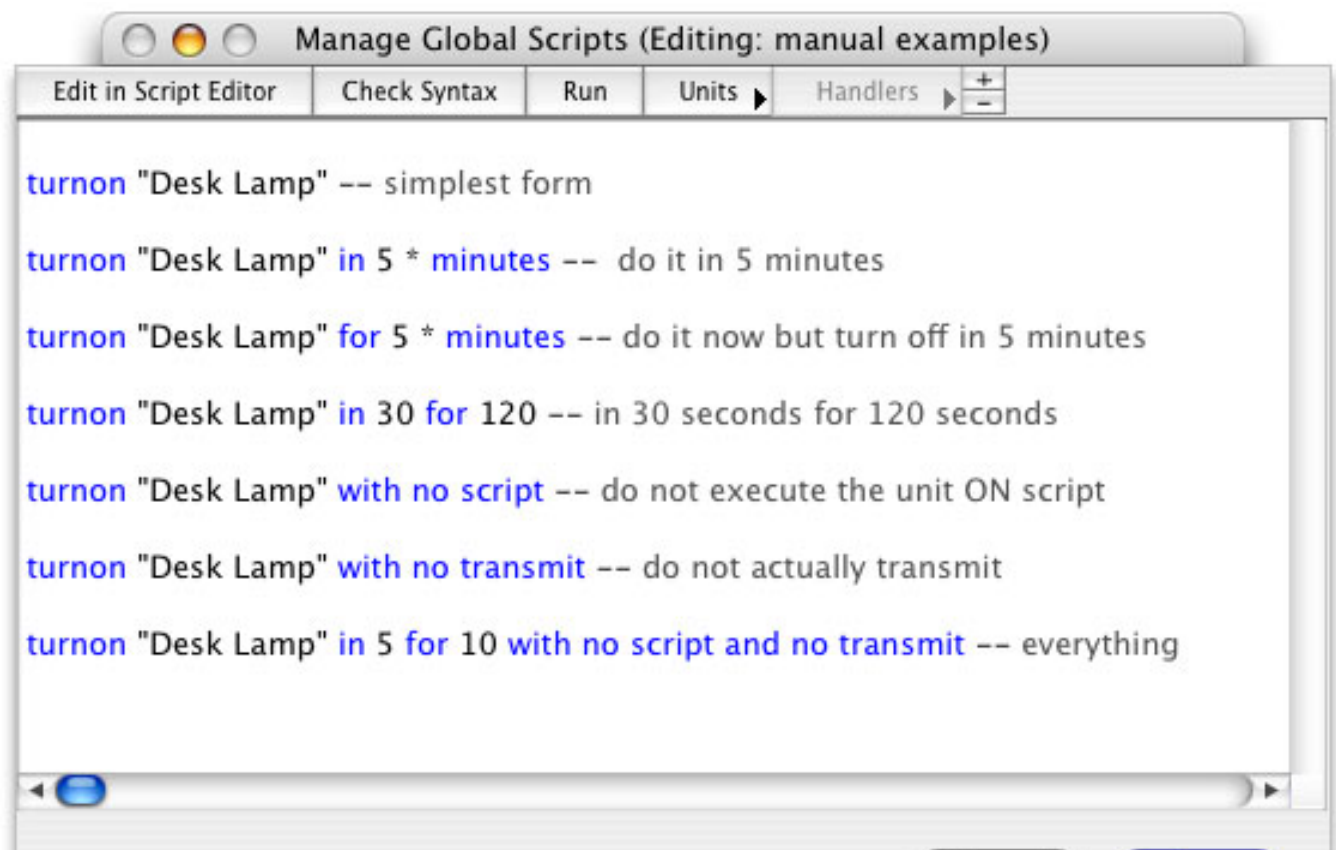


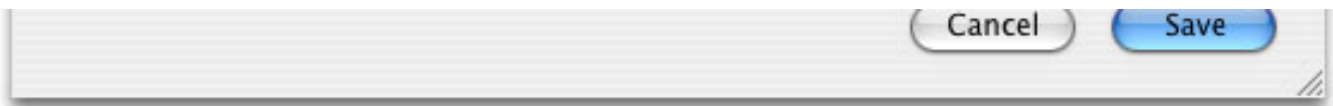
...as well as those that have many different options :



Now, in order to create a script command, just use the parts of the syntax you want, and add the variables including the 'object' of the command... in this case, the unit : Desk Lamp

Do notice that anytime you want to use the name of a XTension Database UNIT, you must enclose it in quotation marks ...





Note that the syntax given in the Script Editor display of the dictionary, as well as the following manual pages, uses the "**Brackets**" to signify that a parameter is **optional**.

Note also, that although you might create something that is difficult to read, there is otherwise no restriction on the sequential ordering of the options.

### Other special names

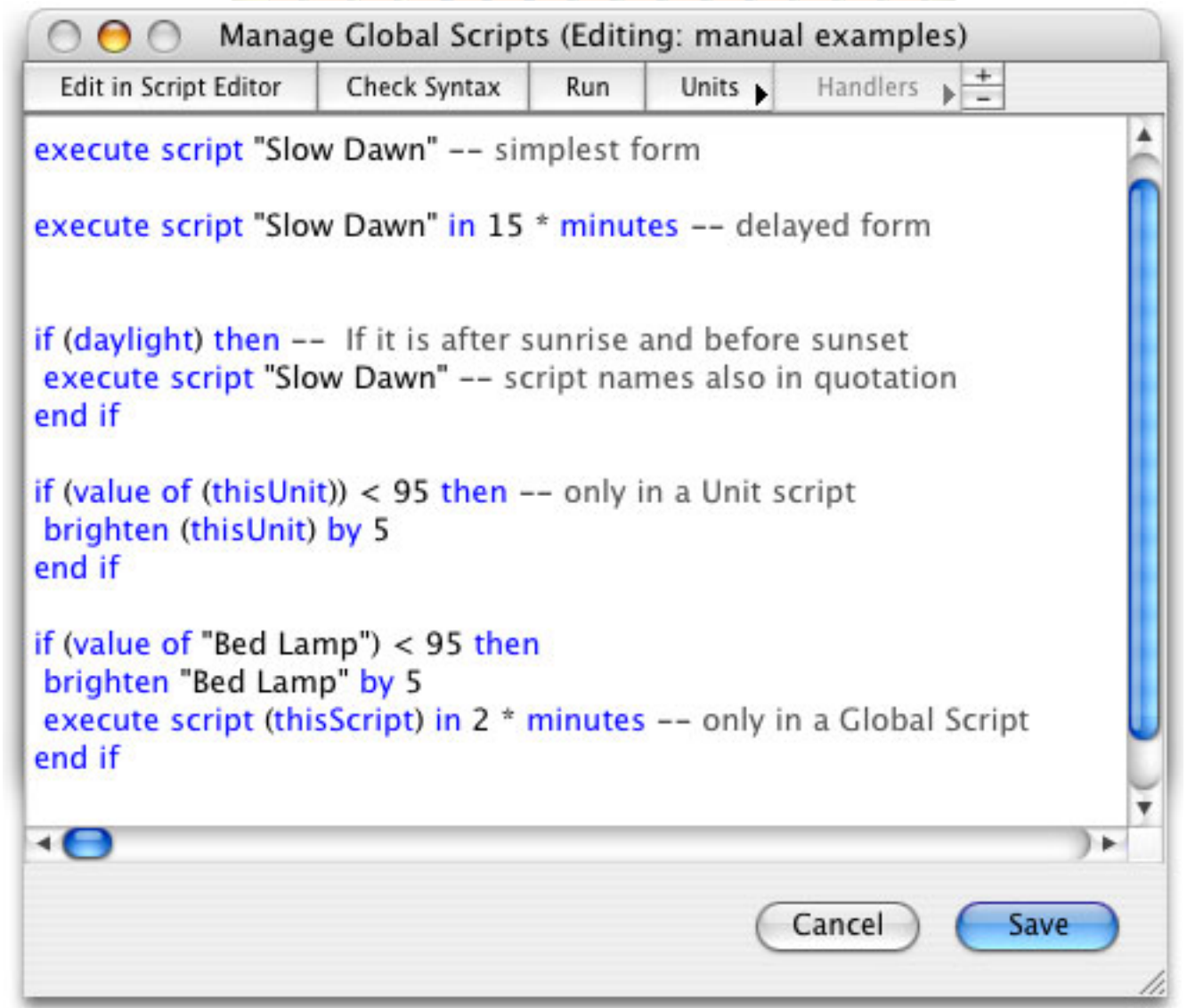
As with any natural language there are certain constructs and 'nouns' that are relative or even self-referential.

AppleScript needs to be able to recognize these absolutely, so we ask for a bit more rigor in the syntax.

It may be that a script is parse-able without the parentheses, but it is definitely recommended.

For example, there is a set of 'nouns' like (**daylight**) that would ordinarily be difficult for the AppleScript parser to recognize if we didn't put them inside of parentheses.

Likewise the self-referential pronouns (**thisUnit**) and (**thisScript**), need to be so formatted.



**Here begins the XTension scripting dictionary**



## Verbs which Turn ON/OFF a unit or group :

### *turnoff or turn off: Command a named unit or group off*

**turnoff** string -- unit or group name

**turn off** string -- unit or group name

[**in** integer] -- turns the unit off after the specified number of seconds

[**for** integer] -- turn unit on after the specified number of seconds

[**with no script**] -- defeats execution of any script for the unit.

[**with no transmit**] -- just don't send the command to the X-10 bus

*example : turn off "All Outside Lights"*

*or: : turn off "All Outside Lights" in 5 \* minutes*

*or: : turn off "All Outside Lights" with no script*

*or: : turn off "All Outside Lights" with no transmit*

*or: : turn off "All Outside Lights" in 10 for 10 with no transmit and no script  
(this last one is just an example...)*

This verb provides the basic method of turning a specific unit or group of units 'OFF'.

This verb does not defeat the 'Block' functions. If you issue a turnoff command to a 'blocked' unit, the unit will not change state.

Note that you can use the '**in**' option here. This option takes a number of seconds as an argument. Thus you can delay the turnoff by a period of seconds ( any number ).

The '**for**' option allows you to specify the amount of time that the unit is to be left off. If used, XTension will automatically turn the unit on after the specified period.

Also, you can use the '**with no script**' option which will defeat the execution of a script which might be attached to the OFF transition of the unit.

The '**with no transmit**' option tells XTension that you do not wish to actually send the command to the X-10 interface/bus. This might be useful when you know that a unit is already ON, the database says that it's OFF, and you don't need to send a X-10 command to the device.

### *turnon or turn on: Command a named unit or group on*

**turnon** string -- unit or group name

**turn on** string -- unit or group name

[**in** integer] -- turns the unit on after the specified number of seconds

[**for** integer] -- turn unit off after the specified number of seconds

[**with no script**] -- defeats execution of any script for the unit.

[**with no transmit**] -- just don't send the command to the X-10 bus

*example : turn on "All Outside Lights" in 5 \* minutes with no script for 5*

This verb provides the basic method of turning a specific unit or group of units 'ON'.

Note that this verb does not defeat the 'Block' functions. If you issue a turnon command to a 'blocked' unit, the unit will not change state.

Note that you can use the '**in**' option here. This option takes a number of seconds as an argument. Thus you can delay the turnon by a period of seconds ( any number ).

The '**for**' option allows you to specify the amount of time that the unit is to be left on. If used, XTension will automatically turn the unit off after the specified period.

Also, you can use the '**with no script**' option which will defeat the execution



of a script which might be attached to the ON transition of the unit.

The '**with no transmit**' option tells XTension that you do not wish to actually send the command to the X-10 interface/bus.

### **toggle: Toggle the state of a unit**

**toggle** string -- unit name

**[in** integer]

**[for** integer] -- toggle unit again after the specified number of seconds

**[with no script]** -- defeats execution of any script for the unit.

**[with no transmit]** -- just don't send the command to the X-10 bus

*example : toggle "Electric Door Latch" in 5 with no script  
(all above examples apply)*

This verb provides the basic method of changing the current state of a specific unit or group of units.

Note that this verb does not defeat the 'Block' functions. If you issue this command to a 'blocked' unit, the unit will not change state.

Note that you can use the '**in**' option here. This option takes a number of seconds as an argument. Thus you can delay the action by a period of seconds ( any number ).

Also, you can use the '**with no script**' option which will defeat the execution of a script which might be attached to the ON or OFF transition of the unit.

The '**for**' option allows you to specify the number of seconds until the unit will be toggled again.

### **turnoff address: Send an "off" command directly to an X-10 address**

**turnoff address** string -- unit, housecode or "all units"

*example : turnoff address "A7"*

This verb provides the basic method of turning a specific X-10 address 'OFF'. Note that this verb is not affected by the 'Block' functions. If you issue a turnoff unit command to a 'blocked' unit, the unit will change state.

### **turnon address: Send an "on" command directly to an X-10 address**

**turnon address** string -- unit, housecode or all

*example : turnon address "J4"*

This verb provides the basic method of turning a specific X-10 address 'ON'. Note that this verb is not affected by the 'Block' functions. If you issue a turnon unit command to a 'blocked' unit, the unit will change state.

### **force on: Force a units state to on, regardless of "blocking" status**

**force on** string -- unit name

**if** integer] -- delay in seconds

**[for** integer] -- force unit off after the specified number of seconds

**[with no script]** -- defeats execution of any script for the unit.

*example : force on "Foyer Chime" for 5*

This verb provides method of turning a device 'ON' regardless of whether the device has been 'Blocked'.

This verb is included for logical completeness and failsafe. This verb does not affect the 'blocked' state of the unit.

See turnon and turnoff for other options.

***force off: Force a unit's state to off, regardless of "blocking" status***

**force off** string -- unit name

[**in** integer] -- *delay in seconds*

[**for** integer] -- *force unit on after the specified number of seconds*

[**with no script**] -- *defeats execution of any script for the unit.*

*example : force off "Foyer Chime"*

This verb provides method of turning a device 'OFF' regardless of whether the device has been 'Blocked'.

This verb is included for logical completeness and failsafe. This verb does not affect the 'blocked' state of the unit.

See turnon and turnoff for other options.

Also see the chapter about "Command Variables"



## Verbs which DIM a unit or group :

Please see the chapter about "Extended Codes" for other methods of dimming and setting the 'preset level' of smart dimmers.

However, even the 'smart' dimmers will respond to the common X-10 DIM/BRI commands.

(SEE related chapters "Preset Dim" and "Simulated Preset Dim")

### *dim or brighten: Set an analog device to a level between 0 and 100*

**dim** string -- unit name

**brighten** string -- unit name

[**to** small integer] integer value between 0 and 100

[**by** small integer] integer value between -100 and +100 (NOT for Groups)

[**in** integer] number of seconds to delay this action

[**for** integer] number of seconds after which to reverse this action

[**with no script**] -- defeats execution of any script for the unit.

[**with no transmit**] -- just don't send the command to the X-10 bus

*example : dim "Foyer Light" to 45*

*example : dim "Foyer Light" to 45 in 30*

*example : dim "Foyer Light" to 45 in 30 for 60*

*example : dim "Foyer Light" by -10 in 30 for 60*

*example : dim "Foyer Light" by -1 in 30 for 60 with no script*

*example : dim "Foyer Light" by -1 in 30 for 60 with no transmit*

*(all above examples apply)*

This verb provides the basic method of dimming a database unit or group of units.

Note that this verb is affected by the 'Block' functions. If you issue a dim command to a 'blocked' unit, XTension will not change its state, and will not execute its scripts.

You may use **either** the "**to**" clause **or** the "**by**", but not both. If the 'by' is present, then the 'to' will be ignored.

The 'by' clause is intended for 'relative' changes, and makes the script look a little more like English:

*brighten "Bathroom Light" by 10*

As an aid in preventing limit errors in such 'relative' changes, XTension will limit the range of the command to values between 0 and 100. Thus, if you try to dim a lamp below 0, XTension will limit it to zero. Likewise, if you try to brighten a lamp above 100, XTension will limit it to 100.

Note that you can use the '**in**' option here. This option takes a number of seconds as an argument. Thus you can delay the dim by a period of seconds ( any number ).

The "**for**" clause is provided so that you can do things like :

*brighten "Bathroom Light" to 50 for 3 \* minutes*

Also, you can use the '**with no script**' option which will defeat the execution of a script which might be attached to the transitions of the unit.

The '**with no transmit**' option tells XTension that you do not wish to actually send the command to the X-10 interface/bus. This might be useful when you know that a unit is already ON, the database says that it's OFF, and you don't need to send a X-10 command to the device.

***dim address: Send Dim or Brighten commands directly to an X-10 address***

**dim address** string -- unit **by** value (negative = Brighten, positive = Dim)

*example : dim address "J4" by 20*

This verb provides the basic method of sending a number of hard DIM/BRI commands to a specific X-10 address.

Note that a Negative Dim = Brighten.

Also see the chapter about "Command Variables"



## Verbs which command a whole 'housecode' :

The X-10 "ALL..." type commands affect all units on a single housecode. They can be issued by your scripts, and by external controllers.

PLEASE NOTE that XTension will change the value of database units that have the same House Code, according to the type of command, and the type of unit.

But that normal **UNIT** scripts are NOT triggered by any of these ALL type commands.

In order to **respond** to incoming ALL commands, you must create global scripts of the name : **DoAll.x** [x = the housecode].

Then, when XTension receives any All type command from the powerlines, it will trigger such a script if it exists.

The DoAll.x script then may test the (**command**), to determine which one of the "All" commands it was.

IF you do **NOT** want the database to be so changed, you may change the reserved variable '**future value**' to any non-zero value. Doing so will tell XTension NOT to make the changes to the database units.

ie: Global Script DoAll.A :

... change future value to 1 -- defeat change of database.

If you want to issue any of these commands **from** your scripts, here are the verbs you'll need :

### **all lights on: Turn on all lights in a housecode**

**all lights on** string -- housecode

example : all lights on "A"

This verb provides the common X-10 function for turning on all lamp modules on a single housecode.

### **all lights off: Turn off all lights on a house code**

**all lights off** string -- housecode

example : all lights off "A"

This verb provides the common X-10 function for turning off all lamp modules on a single housecode.

Please note that not all dimmable units will respond to this command.

### **all units off: Turn off all units on a house code**

**all units off** string -- housecode

example : all units off "A"

This verb provides the common X-10 function for turning off all modules on a single housecode. Note that some devices do not respond this command.



## "Stacking" of X-10 addresses before a command:

Within the X-10 protocol, it is possible to 'select' several unit addresses which are on the Same Housecode, and then send one or more commands that will affect all of the selected units at the same time.

This allows you to do things like simultaneous control of a set of lights.

### *send only address : a unit by address*

**send only address** string -- Valid X-10 address [A-P] [1-16]

*example : send only address "A6"*

*send only address "A7"*

*turnon address "A8" --- all three units come on at once.*

### *send address for: a unit by unit name*

**send address for** string -- NAME of Unit in the XTension Database

*example : send address for "Dining Lamp 1"*

*send address for "Dining Lamp 2"*

*turn on "Dining Lamp 3" --- all three units come on at once.*

This verb is fully functional after the release of XTension 3.0.9

The last unit will be updated in the database, but the first two will not be. Thus the 'current values' of some units will not be correct.

If you control unit values directly, without depending on their current value, you shouldn't have a problem creating very nicely coordinated 'Scenes'...

The decision to include this verb was based on the demand for features that could assist in the programming of PCS (brand) dimmers which require that we be able to send 'addresses' separate from 'commands'.

Other things that don't work right yet :

You cannot specify the name of a 'Group'... you will get an error message

No unit scripts will be executed.

## Using units as counters or values :

### Pseudos?

Sometimes you need to have a database unit that doesn't represent some actual X-10 device, but rather is just a place to put some value like 'temperature', or count things like 'number of times that the door was opened today'...

### *set value of : set the value of a unit as a counter*

**set value of** string ( *Name of database unit* )

[**to** integer] integer value between -2,147,483,648 and +2,147,483,648. (32 bits )

[**by** integer] integer value between -2,147,483,648 and +2,147,483,648 (32 bits )

[**with no script**] -- *defeats execution of any script for the unit.*

*example : set value of "Temperature" to 79*

*example : set value of "Door Entries" by 1*

*example : set value of "Door Entries" by -45*

This verb allows you to modify counters and values which are not limited by the 0-100 range of lamp modules (which have an X-10 address).

NOTE : if you use this verb with one of your units that has an assigned 'address', then it is assumed that the value can only be 0-100.

### *value of: Get the analog value of a unit*

**value of** string -- *unit name*

Result: small integer

*example : if value of "Temperature" is greater than 70 then  
    turnoff "Heating"  
end if*

This verb returns the analog value of a unit. If the unit is 'dimmable', then this value can be set via the 'DIM' verb, or through the control panel. The value can range from 0 to 100.

### So What ?

Using this verb, you can create counters for such things as rainfall, total light hours, etc. Obviously, using other AppleScript operators is legal, so you can do things like :

set value of "Daily Rainfall" to ((value of "Daily Rainfall") + (value of "Hourly Rainfall"))

## Verbs which get information about a unit or group :

Please note that although you can specify a unit which is a Group, the values returned by these verbs reflect the results of changes to the Group specifically.

So that if any member of the Group changes, that does not cause a change to the unit that represents the Group.

There are also some special status verbs for the 'smart' dimmers that require the 'extended code' type commands. See the chapter on [Extended Codes](#).

### **time delta of:** Returns number of seconds since the named unit changed state

**time delta of** string -- unit name

Result: integer -- in seconds

example : if time delta of "Garage Motion" > 3 \* minutes then  
    turnoff "Garage Lights"  
end if

Every time XTension changes the value of a database item, it also changes the last time stamp for the unit.

This verb will return the number of seconds since the named unit or group was last changed by XTension, whether from autonomous input or by a script. IE: time of last change to database for that unit/group.

### **last timestamp of:** Get the last timestamp for a unit

**last timestamp of** string -- unit name

Result: long date -- timestamp in Apple long date format

example : write log "The door bell last rang at " & last timestamp of "Door Bell"

This verb returns the value of the last timestamp for the given unit or group. IE: time of last change to database for that unit/group.

### **status of:** Get the status of a unit

**status of** string -- unit name

Result: 'unen' -- unit status

example : if status of "Garage Motion" is true then  
    turnon "Outside Siren"  
end if

This verb provides the basic function of getting the current status of any unit or group in the database.

The result is a value which can be tested for true/false.

False = 0 , True ≠ 0

### **set state of:** Set the status of a unit

**set state of** string [onState/offState]

example : set state of "Garage Motion" to onState

This verb provides the basic function of setting the current status of any unit or group in the database.

This should only be used with On/Off type units. see set value for 'analog' types.

This verb will not cause either of the Unit Scripts to be called.

### **value of:** Get the analog value of a unit

**value of** string -- unit name

Result: small integer

example : if value of "Temperature" is greater than 70 then  
    turnoff "Heating"  
end if

This verb returns the analog value of a unit. If the unit is 'dimmable', then this

value can be set via the 'DIM' verb, or through the control panel. The value can range from 0 to 100.

**set value of: Set the current value of a unit**

**set value of** string to integer [with no script]

*example : set value of "Rainfall" to 55 with no script*

This verb provides the basic function of setting the current value of any unit in the database.

This should only be used with Analog (dimnable) type units. see set status for 'on/off' types.

**query of: Send a status request to a unit and receive status response**

**query of** string

**Result:** small integer

*example : if query of "Driveway Motion" > 0 then  
write log "Someone is moving at the front driveway sensor"  
end if*

This verb provides a method of issuing the X-10 standard 'status request' to devices which respond to such commands. Recent devices have been developed which will respond to this command with the current value of the unit's state. At the current time, only discrete values are returned from any unit which honors this X-10 command. This means only True/False can be tested.

Also see the chapter about "Command Variables"



## Verbs for Blocking :

### **block unit: Block a unit from being activated**

**block unit** string -- Unit name  
[**in** integer] -- seconds until unit is blocked  
[**for** integer] -- seconds to leave unit blocked  
example : block unit "Outside Siren"  
example : block unit "Outside Siren" in 5 \* minutes  
example : block unit "Outside Siren" for 5 \* minutes  
example : block unit "Outside Siren" in 5 \* minutes for 5 \* minutes

This verb allows a script to prevent any other script from issuing a command to either a single unit/address, or a group of units. Thus, scripts can be written for sensors which always turn on lights or sound alarms regardless of other conditions. You then write scripts which block or unblock units or groups according to time of day, daylight, password entry, etc.

Please note that there is an option in the unit setup dialog which allows you to make any unit to be 'Receive only'. This a semi-permanent way of preventing any script from changing some unit. The 'block unit' and 'unblock unit' verbs provide a way of dynamically guarding a unit or group of units.

Note that the 'force ..' verbs are specifically provided to override the blocking function.

### **unblock unit: Remove block from a unit**

**unblock unit** string -- Unit name  
[**in** integer] -- seconds until unit is unblocked  
[**for** integer] -- seconds to leave unit unblocked  
example : unblock unit "Outside Siren"  
example : unblock unit "Outside Siren" in 5 \* minutes  
example : unblock unit "Outside Siren" for 5 \* minutes  
example : unblock unit "Outside Siren" in 5 \* minutes for 5 \* minutes

This verb provides the reverse of the 'block unit' command. It can be used to unblock a unit or group.

### **remove all blocks: Remove blocks from all database units**

**remove all blocks** -- no arguments

This verb will unconditionally remove all blocks on all database units. It is included for failsafe and logical completeness of the verb set.



## Scheduled Event Verbs :

### **execute script:** Execute a global script

**execute script** string -- script name

**[in** integer] -- delay in seconds

example : execute script "Call 911" in 3 \* minutes

This verb allows any script to execute any other script. Note that either the script or an alias for the script must be in the 'Scripts' folder in the XTension home folder.

As with some other verbs, you may use the delayed option : "IN"

### **create event:** Create a scheduled event

**create event** string -- *Event name in Quotes* or Null (XTension will create name)

**that** turnson/turnsoff/toggles/executes/dims/presets/blocks/unblocks

**unit** string -- (or **script** string if verb is 'execute')

**to** **[level** small integer]

**in** **[integer]** (seconds)

**starts at** date] (AppleScript type: date)

**repeats every** integer] (seconds, Applescript: \* Minutes,Hours,Days)

**randomize by** small integer] (seconds only)

**with no script**] (just don't execute any Unit scripts)

**[weekdays** string] -- Use a 7-character string of this form: "-M-W-F-" Use dashes for days not to execute.

example : create event "Morning Coffee" that turnson unit "Coffee Pot" starts at (date) repeats every 1 \* days weekdays "-MTWTF-"

example : create event "Evening Tea" that turnson unit "Tea Pot" starts at ((sunset for the (current date)) - 20 \* minutes) weekdays "-MTWTF-"

example : create event "Nighty" that executes script "Going to Bed" in 15 \* minutes

This verb will create a new scheduled event with explicit start time and repeat period. You can turn on/off, dim or toggle a unit, or execute a script. You can even randomize the event, and choose the weekdays.

Please note that a quirk of this verb for executing a script requires that you say : create event .... that executes **unit** ["script name"].

### **suspend event:** Suspend a scheduled event

**suspend event** string -- *Event name or Unit name in quotes*

**[for** integer] -- seconds until 'next event time'

**[until** date] -- date and time for next event time

example : suspend event "Morning Coffee" for 2 \* minutes

example : suspend event "Coffee Pot" for 1 \* hours

### **unsuspend event:** unSuspend a scheduled event

**unsuspend event** string -- *Event name in quotes*

These two verbs allow you to temporarily suspend a scheduled event. The event remains in the event list, but it just does not execute at its scheduled time if it is suspended.

Of course, you need to know the Event Name to do so.

Thus, for such events, you must assign Names to them if you expect to be able to suspend/unsuspend them.

### **remove event:** Remove a specific scheduled event

**remove event** string -- *Event name*

example : remove event "Morning Coffee"

**[for unit** string] -- removes next event for this unit

This verb will remove the **next** scheduled event for either the named event or for the named unit.

**[remove unit events:](#)** ***Removes all scheduled events for a unit***

**remove unit events** string -- unit name  
*example : remove unit events "Coffee Pot"*

This verb will remove **all** scheduled events for the device or group.

**[all events:](#)** ***Returns a text list of the names of all scheduled events***

**all events**  
*example : set EventList to all events*  
*result : list of the names of all events*

This verb simply returns a standard list of the names of all scheduled events.



## Human Interface Verbs :

### **write log:** Write a timestamped message to the log file

**write log** string -- message to log

[**color** black/blue/green/red] -- Set color of the text to : Black, Blue, Green, Red

example : write log "Motion detected in the garage"

This verb provides the ability of any script to write a timestamped message to the log file, as well as a Colored entry to the Log Window.

PLEASE NOTE: When you specify the color BLACK: This is a special method of specifying that you do not want this line to appear in the LOG WINDOW when you have elected to display "Exceptions Only". It will of course always be written to the LOG FILE.

### **display dialog:** Display a dialog for the user

**display dialog** string -- text to display

example : display dialog "There is a Fire in the Basement"

NOTE: this special verb creates a NON-modal dialog --it won't stop XTension

### **say:** Use the Macintosh speech synthesizer to speak a message

**say** string -- Message to speak

[**in** integer] -- turns the unit off after the specified number of seconds

example : say "There is a FIRE in the GARAGE"

example : say "Take the roast out of the oven !" in 60 \* minutes

This verb will call upon the Macintosh speech module to annunciate the given string.

NOTE: this special verb creates a NON-modal dialog --it won't stop XTension

### **put picture** Put a PICT image from the Views folder into a View

**put picture** string **in view** string

example : put picture "Evening Bedroom" in view "Bedroom View"

This verb assumes that you have created a folder called "Views", in the same folder with XTension. The items in that folder should be PICT images and can be aliases.

Note that this verb does NOT change the name of the View window. You can change it manually if you want, but the idea is that you want to have a selectable set of images to put in a View window, and you don't want that window name changing. (hard to script)

### **front window:** Bring a specified window to the front

**front window** string -- window name

example : front window "Master List"

This verb is used to bring one of the XTension windows to the front. Used in conjunction with the highlight unit verb, it can be used to point out a specific unit in any View or List.

### **hide window:** Hide a named window if it is visible

**hide window** string -- window name

example : hide window "Master List"

This verb is used in conjunction with the 'front window' verb for just the obvious reason.

### **set icons in:** Select the size of icons in the named View

**set icons in** string -- window name

**to** string small/medium/large

example : set icons in "House View" to small

This verb give you a scriptable method of changing the overall icon size for all icons in a named "View".

**highlight unit: Selects a unit in the front window**

**highlight unit** string -- unit name  
example : front window "Master List"  
highlight unit "Foyer Light"

This verb will cause any database unit in the front window to be selected. In a View, the icon for the unit is selected, and the name is displayed below the icon.

**set on icon for: Select a named icon as the ON icon for a unit**

**set on icon for** string -- unit name  
**to string** -- name of icon in XTension Icons file  
example : set on icon for "Foyer Light" to "Lamp Bright"

**set off icon for: Select a named icon as the OFF icon for a unit**

**set off icon for** string -- unit name  
**to string** -- name of icon in XTension Icons file  
example : set off icon for "Foyer Light" to "Lamp Dimmed"

These verbs will change the ON/OFF icon for any database unit to the icon specified by the 'to' string. These icons must be in the XTension Icons file, and each must be 'named'.

There is a new XTension Icons file that gives names to all of the basic icons. It can be downloaded from the shed.com site, and is available with the CD and all versions of XTension after 3.0.4.

You can edit the file and change the names, and add icons, using a resource editor or icon editor like Iconographer.

**place icon for: Put the icon for a unit into a 'View'**

**place icon for** string **horiz** [integer] **vert** [integer] **in view** string  
example : place icon for "Foyer Light" horiz 10 vert 10 in view "Living Room"  
this will put the icon for "Foyer Light" into the named graphic view at pixel location x10,y10.

This verb will place a unit icon at the specified location, in the named graphic View. The view must be specified.

Note that the pixel offset is relative to the origin of the window.. [0,0]  
It will only work with 'Views'.

**remove icon for: Put the icon for a unit into the front 'View'**

**remove icon for** string **from view** string  
example :remove unit "Foyer Light" from view "Front View"

This verb can be used in conjunction with the place icon verb. it simply removes the icon for a unit from the selected View.

## **Logging Control Verbs :**

These verbs will help when you need to control the current logging mode from within Scripts:

### **set logging: Sets the level of detail displayed in the Log Window**

**set logging** string -- "*all*" "*normal*" or "*exceptions*"  
*example : set logging "all"*

This verb will change the current logging detail preferences to the desired level of detail. "All" of course will cause every event to be written to the Log Window, and "exceptions" will limit the detail to only errors or explicit 'write log' events.

### **log mode: Returns the current logging mode**

**log mode** -- result = "*all*" "*normal*" or "*exceptions*"

This verb simply returns the current text string equal to the current log mode set in the Preferences, manually, or from the 'set logging' verb.

### **cut log file: Force close and reopen of Log File**

**cut log file**

Use this verb to close the current Log File, and start a new one. Generally useful for limiting the size of log files.

### **set disklogging: Enable or Disable Disk Logging**

**set disklogging** [true/false]

This verb should only be necessary when disk space is depleting, or after it has depleted and has been corrected.

When XTension discovers that disk space for the disk logs has depleted, it will stop logging to disk. Whenever XTension is restarted, disk logging will be resumed unless the disk space is still depleted.

If you want to take action automatically to the disk is full condition, then you can provide a special Global Script named "Errors Script".

There you can either compress the old Log files, or delete them. After you have taken action to correct the condition, you can then execute this verb in order to re-establish logging to disk.



## Special verbs for editing the database from external applications (or anywhere)

There are many applications that can either provide special human interfaces, or that simply need to be able to 'tell' XTension to 'do things' to items in the database.

Also see chapter for Ad Hoc Unit Properties.

### create unit/group: Create a new unit or group in the database

**create unit** string **address** string (or null)

**create group** string

example :

**create unit** "Lamp" **address** "A1" -- create a 'real' unit

**create unit** "High Temp" **address** "" -- create a 'pseudo' unit

**create group** "Front Lights" -- create a new group

Note that this verb does not allow duplicate names, and will only allow duplicate addresses according to the user preferences.

### delete unit/group: Delete a named unit from the database

**delete unit** string

**delete group** string

example :

**delete unit** "Lamp" -- delete a unit (or group)

**delete group** "Front Lights" -- delete a group

Note that the 'delete group' verb is provided only for script clarity.

### set description of unit: Modify the description field of a unit

**set description of** string **to** string (or null)

example :

**set description of** "Lamp" **to** string -- set the description field of a unit

(Works for groups too.) Note that the Description field is limited to 254 characters, but it can be used for any purpose, from simple text descriptions, to things like 'recent history' of this unit...

### description of unit: Retrieve the description of a unit

**description of** string

example :

**set theDescription to description of** "Lamp" -- get the description into a variable  
(Works for groups too.)

### set params of unit: Set the unit type variables for a named unit

**set params of** string [**Param** option]...[]...[]...

examples :

**set params of** "Lamp" dimmable true simulate true single false

or:

**set params of** "Lamp" with dimmable with simulate without single  
(these two statements do exactly the same thing...)

**Variables : dimmable, simulate, smart, reverse, receive, single, wireless, passthru**

(All applicable items work for groups too.)

### set name of : Rename a unit

**set name of** string (database unit name)

**to** string (new database unit name)

This verb allows you to change the name of a unit from within a script. It is a **dangerous** verb in that you can use it to really do some nasty things to yourself.

If you don't know that you need it, don't use it.

### **set address of : Change the address of a unit**

**set address of** string (database unit name)  
**to** string (house code/unit)

This verb allows you to change the address of a unit from within a script. It is a **dangerous** verb in that you can use it to really do some nasty things to yourself. If you don't know that you need it, don't use it.

### **unit address of : Get the address of a unit**

**unit address of** string  
returns: string

This verb allows you to get the address of a unit from within a script. It returns a text string of TWO or THREE characters. ie: A1 to P16

### **add unit to group Add a unit to a named group**

**add unit** string **to group** string -- Name of Database Unit and name of Group

### **remove unit from group Remove a unit from a named group**

**remove unit** string **from group** string -- Name of Database Unit and name of Group

examples :

**add unit** "Front Door Light" **to group** "Night Lights"  
**remove unit** "Front Door Light" **from group** "Night Lights"



## Special verbs getting Lists of units [with extract]

There are many applications that can either provide special human interfaces, or that simply need to be able to 'tell' XTension to 'do things' to items in the database.

Most of these verbs return an AppleScript 'list'. Normal use is to simply get the names of Units, verbs etc. But they may also be used to extract database fields along with the name.

Please note that the explanation of 'with extract' is in the next chapter :

### unit type: Return a 'list' of all units of a specified 'type'

**unit type** string -- all, real, dimmable, smart, analog, discrete, pseudo, blocked, wireless, passThru  
[with extract] -- returns Unit record values (see "Extract" chapter...next)

example :

```
set aList to "" as list
set aList to unit type "analog"
```

```
write log "we got this many analog units: " & (count of items of aList)
write log "and here's their names and current value:"
repeat with x from 1 to count of items of aList
  set u to item x of aList
  write log " : " & u & (value of u)color green
end repeat
```

### all of class: Return a 'list' of all verbs, globals, lists, views or groups

**all of class** string -- Verbs, Globals, Lists, Groups, Views  
[with extract] -- returns Unit record values (see "Extract" chapter...next)

example :

```
set aList to "" as list
set aList to all of class "Globals"
write log "we got this many global scripts: " & (count of items of aList)

repeat with x from 1 to count of items of aList
  set z to item x of aList
  write log "Like this : " & z
end repeat
```

Thus, you can gain access to all of the global scripts, all of the verbs that XTension serves, and the names of all of the Groups in your database.

### all of view: Return a 'list' of all units in a specified 'View'

**all of view** string -- name of List,

### all of list: Return a 'list' of all units in a specified 'List'

**all of list** string -- name of List,  
[with extract] -- returns Unit record values (see "Extract" chapter...next)

example :

```
set aList to "" as list
set uList to "" as list
set aList to all of class "Lists"
write log "we got this many Lists: " & (count of items of aList)

repeat with x from 1 to count of items of aList
  set g to item x of aList
  write log "In list : " & g & " we have these units : "
  set uList to all of list g
  repeat with z from 1 to count of items of uList
    set u to item z of uList
    write log " : " & u
  end repeat
end repeat
write log "Last unit of last list is : " & last item of uList
```

### all of group: Return a 'list' of all units in a specified 'Group'

**all of group** string -- name of Group,

**[with extract]** -- returns Unit record values (see "Extract" chapter...next)

example :  
**set** aList **to** "" **as** list  
**set** uList **to** "" **as** list  
**set** aList **to** all of class "Groups"  
write log "we got this many Groups: " & (count **of** items **of** aList)

**repeat with** x **from** 1 **to** count **of** items **of** aList  
  **set** g **to** item x **of** aList  
  write log "In group : " & g & " we have these units : "  
  **set** uList **to** all of group g  
  **repeat with** z **from** 1 **to** count **of** items **of** uList  
    **set** u **to** item z **of** uList  
    write log " : " & u  
  **end repeat**  
**end repeat**  
write log "Last unit of last group is : " & **last** item **of** uList

## **all events: Returns a text list of the names of all scheduled events**

### **all events**

**[with extract]** -- returns Unit record values (see "Extract" chapter...next)

example : set EventList to all events  
  result : list of the names of all events

This verb simply returns a standard list of the names of all scheduled events.



## **XTension can return 'record data extracts'**

For most of the 'List Processing' verbs, there is an option that not only returns a list of names, but also returns a 'record' of various elements taken directly from the XTension Database Unit Record. (or Event Record)

For example. Normally if you use the command : unit type "blocked", you would get a simple Text list of all Units that are 'Blocked'.

However, if you append the option 'with extract', to the verb :

**set aList to all units "blocked" with extract**

You would get a List of Records, each with named elements.

**This record format was developed for use by external applications such as 'web interface' apps, but they can be used in common scripts.**

There are basically two different Record types currently returned. One for any XTension Database Unit, and a second for any Scheduled Event.

**The 'named' fields of these records :**

### **Unit record Extract: what you receive for all 'Units' :**

**xtName** : string, Name of the Unit  
**xtValue** : integer, Current Value of the Unit  
**xtFlags** : integer, Boolean processing flags \*  
**xtAddress** : string, Two or three character X-10 address (or null)  
**xtTimestamp** : datetime, old style Date/time long word  
**xtDescription** : string, Text Description of the Unit

**\* Note: versions of XTension after 3.8.0 include Boolean items that may be tested individually. See the 'flags' table below. \*\***

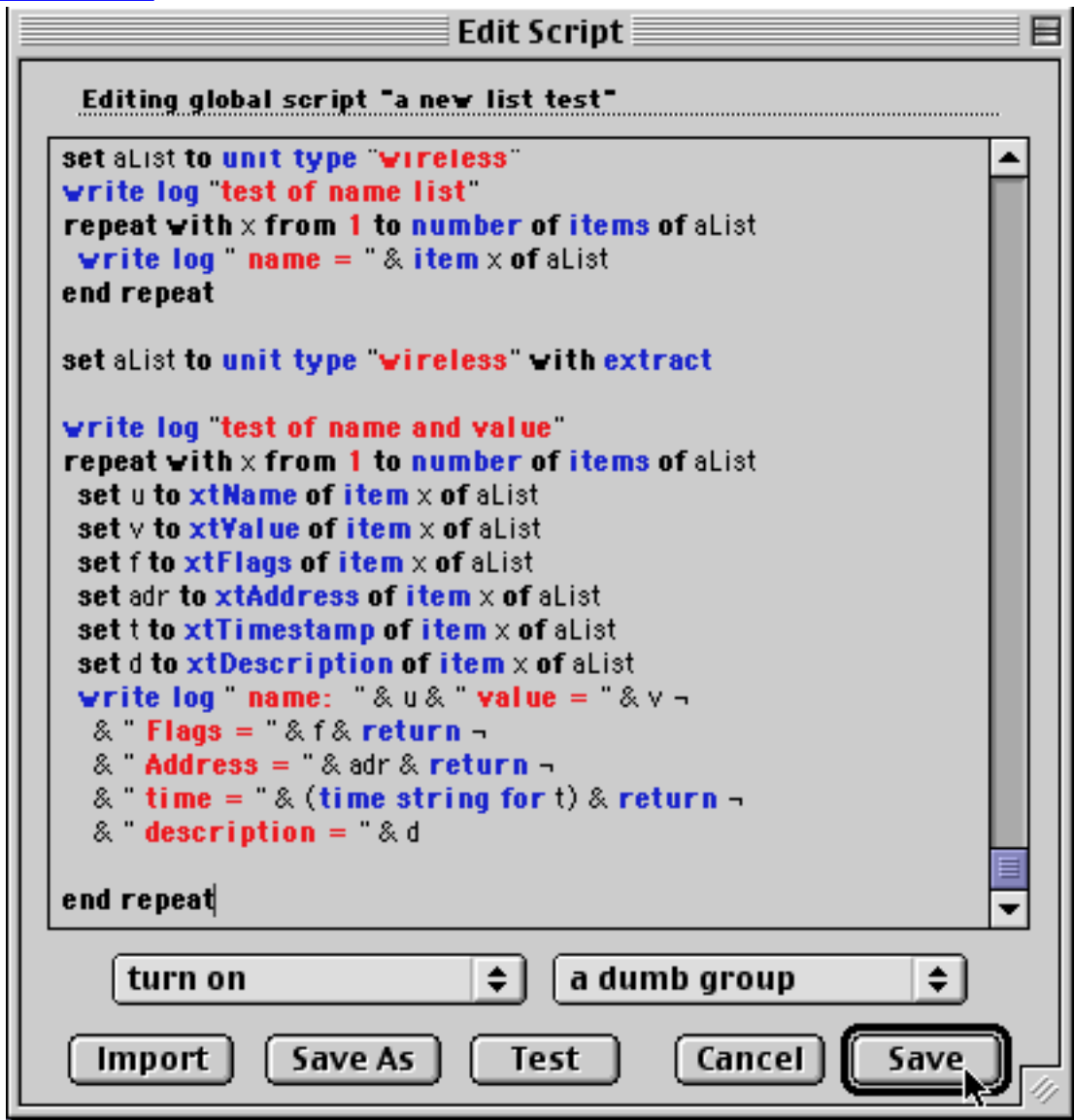
### **Event record Extract: what you receive for all 'Events' :**

**xtEventName** : string, Name of the Unit  
**xtEventAction** : integer, An action code \*(see below)  
**xtEventTime** : datetime, old style Date/time long word  
**xtEventFlags** : integer, Boolean processing flags \*  
**xtEventRepeater** : boolean, Flag- True says this is a 'Repeater'  
**xtEventReps** : integer, Repeat repetition count (\* per minute...)  
**xtEventPeriod** : integer, Coded value of seconds, minutes, hours ...\*  
**xtEventWeek** : integer, Bits represent all or any days of the week\*\*  
**xtEventLevel** : integer, Value to which a dimmable unit will be set.  
**xtEventUnitName** : string, Text Description of the Unit

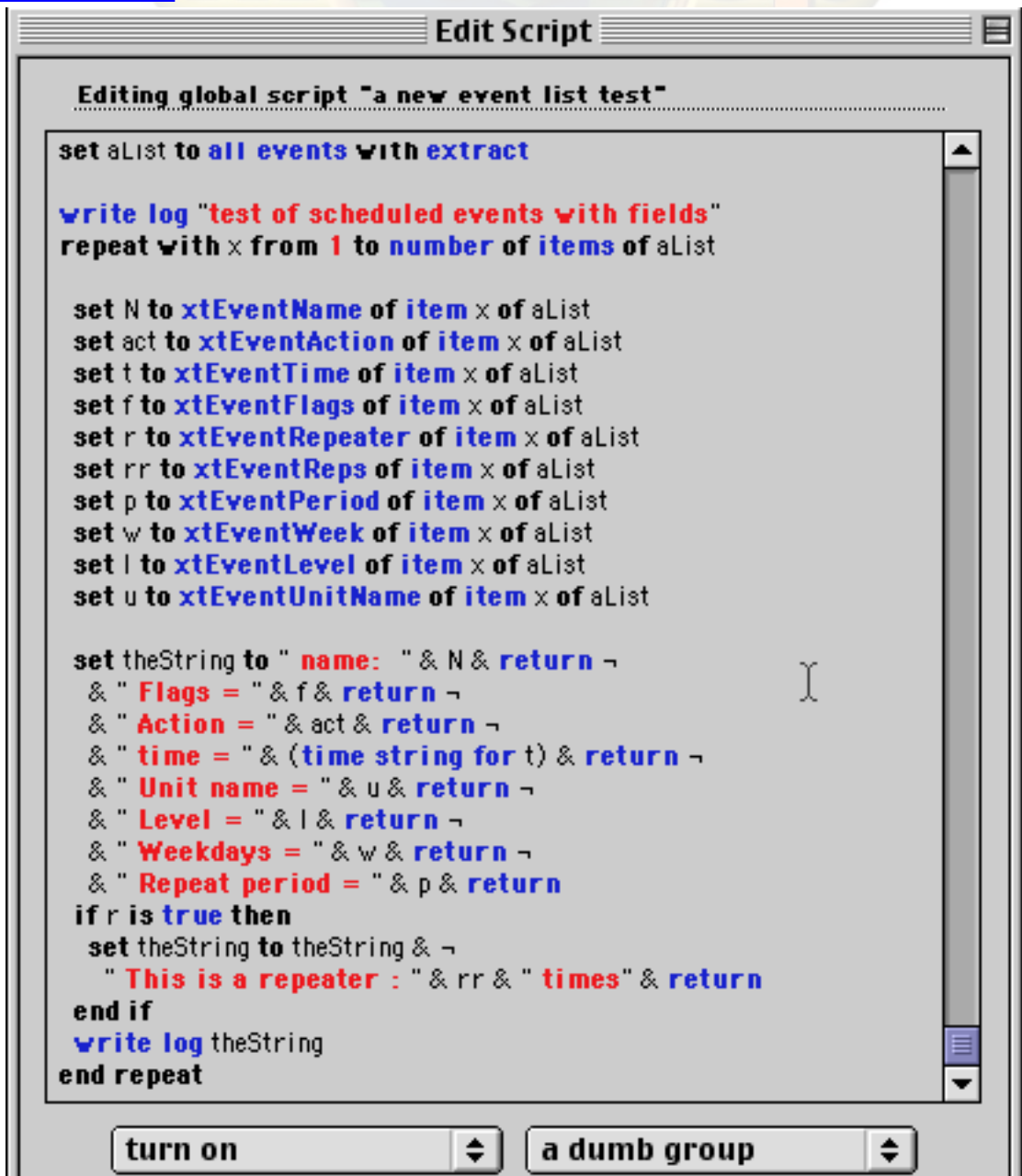


Here are two examples of scripts that use the Record formats:

For "Units":



For "Events":





## \*\* Interpretation of the different Flags and Codes :

### The **Unit** Boolean Properties Word : **xtFlags** :

- Bit 15** = this entry is valid -- tautological -- useless
- Bit 14** = this entry has an X10 address **xtHasAddress**
- Bit 13** = this entry is blocked, do not execute **xtBlocked**
- Bit 12** = this entry is a group **xtGroup**
- Bit 11** = Use reverse logic **xtReversed**
- Bit 10** = This unit accepts dim commands **xtDimmable**
- Bit 9** = Receive Only -- Cannot send commands to this Unit (**xtReceiveOnly**)
- Bit 8** = Simulated preset dim Unit **xtSimPreset**
- Bit 7** = this (Dimmable) unit has been sent an OFF command (**xtHardOff**)
- Bit 6** = this is a real Analog port **xtAnalog**
- Bit 5** = single click in View means Toggle the state **xtSingleClick**
- Bit 4** = This is a Smart Dimmer (self-manages the preset dim) (**xtSmart**)
- Bit 3** = MR26 input allowed ? **xtWireless**
- Bit 2** = Auto-Pass-Thru ? (only if wireless) **xtPassThru**

*Versions of XTension after 3.8.0 put these flags into the extract as individual boolean items. Their names are in green above.*

### The **Event** Boolean Processing Flags : **xtEventFlags** :

- Bit 2** = Event will not cause Transmit of X-10 command
- Bit 1** = This Event is Suspended

### The **Event** Action Code : **xtEventAction** :

The type of event that will occur:

- 3** = Turn ON
- 4** = Turn OFF
- 5** = Dim/Bri
- 33** = Toggle
- 34** = Execute (script)
- 44** = Block
- 45** = Unblock
- 46** = Preset to level

### The **Event** Period Code : **xtEventPeriod** :

The Period of the event.. every x :

- 1** = Seconds
- 2** = Minutes
- 3** = Hours
- 4** = Days
- 5** = Weeks
- 6** = Months

- 7 = Years**
- 8 = Every Sunrise**
- 9 = Every Sunset**

**The Event Day of Week : xtEventWeek :**

The Days of the Week when Events May happen:  
This is a bit-coded word. Only the upper 7 bits of the lower Byte are significant.

- 7 = Sunday**
- 6 = Monday**
- 5 = Tuesday**
- 4 = Wednesday**
- 3 = Thursday**
- 2 = Friday**
- 1 = Saturday**
- 0 = Not used**



## Special X-10 commands :

### Preset DIM:

Within the X-10 set of commands, there is a special command which can be used to extend the capabilities of 'smart' devices which communicate via the powerlines. It was originally intended to be used as a method of setting a 'preset dim' level for lamp dimmers.

There are also features that help to make the old 'dumb' dimmers appear to behave like the 'smart' new dimmers with internal 'preset dim'.

In recent years, it has been discovered by developers to be a reasonable way of passing parameters between 'smart' devices on the X-10 bus.

One of these devices is the RCS Thermostat which uses the preset dim command both to set special variables in the thermostat, as well as for sending the current status of the thermostat back to XTension.

Basically, the command is sent to a specific address. Attached to it is a value of between 0 and 31.

You can issue these commands from your scripts, and XTension will automatically store the latest received 'preset' value in the database.

You can then test that value in your scripts.

### *preset or preset address Command a named unit or specific address*

**preset** string -- unit name

**preset address** string -- X-10 housecode/unit ie: "B5"

[**to level** integer] -- specifies the level from 0 to 31

*example : preset "RCS Temp Setting" to level 25*

### *preset of (unit) -- Returns the current 'Preset' value for a specific named unit*

**preset of** string -- unit name

*example : write log "the current preset for RCS Temp is :" & preset of "RCS Temp Setting"*

Watch for things about this command on the website, and on the XTension discussion list. There will likely be some 'plug-ins' that will make a particular product 'easy' to integrate into your XTension system.

(Note that the 'preset of' verb will also return the 'simulated preset' level for a Simulated Preset Dim type of Unit...)

**Simulated Preset Dim:**

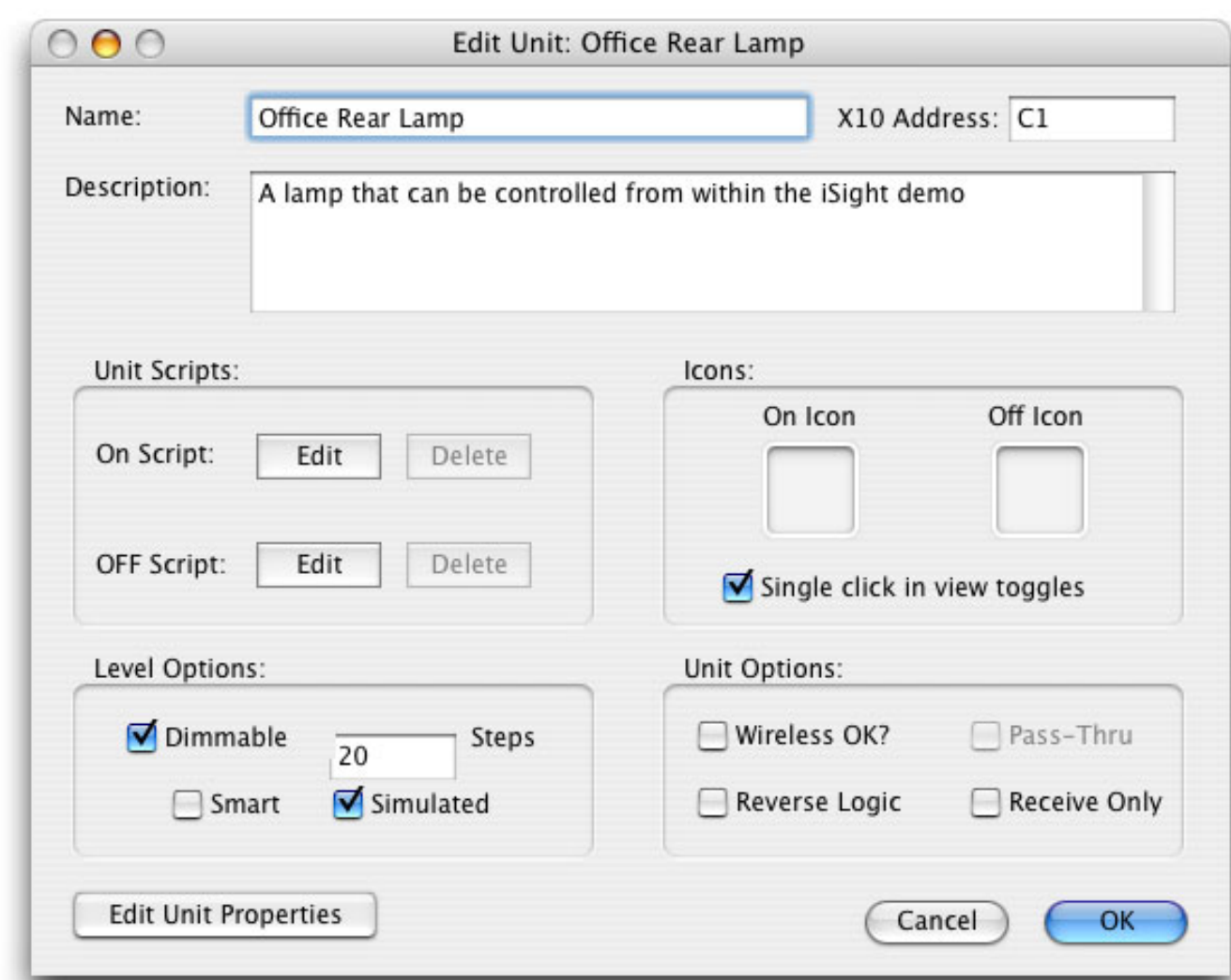
The previous chapter refers to a feature of 'smart' dimmers that can preserve a 'preferential dim level' internally, and with a single ON command will brighten to that level without having to receive multiple Brighten commands.

However, XTension offers a feature which will simulate the same effect with even the 'dumb' dimmers that are most commonly available.

In general, XTension remembers this 'preset level' in the database, with the other Unit variables, and provides methods of changing this preset level.

NOTE: You should visit the shed.com site and study the tutorial:  
<<http://www.shed.com/tutor/justlites.html>>

In order to set up such a Unit, you simply set the options in the Edit Unit dialog : Set Dimmable, and Simulated



**How to manage a Simulated Preset Dim Unit :**

You have three different methods of setting the 'preset level'.

- You can specifically use DIM and BRIGHTEN commands in any script.
- You can manually adjust the SLIDER in the Control Panel
- You can use the "**sim preset**" verb.

With the first two methods, the database is updated immediately, AND the Unit is sent the proper number of DIM/BRI commands to change it to the desired level of brightness.



With the '**sim preset**' verb, ONLY the database is changed.

IE: no X-10 command will be sent.

***preset or preset address Command a named unit or specific address***

**sim preset** string -- unit or group name

**to** integer -- specifies the level from 0 to 100

*example : sim preset "Office Lights" to 25*

Once the 'preset level' is set, all you do from then on to control the Unit, is to use the ON and OFF commands.

XTension will always DIM the Unit to Zero for an OFF command, and will Brighten the Unit to the Preset level for an ON command.

The idea is that YOU determine what your preference for the level is at different times of day (MODE of the HOME ?), and thereafter, all scripts that control that Unit, need only issue ON or OFF commands.

This saves a lot of time and scripting, and turns those old Dumb dimmers into reasonably well-behaved 'almost smart' dimmers.



## **Extended Codes:**

The 'Extended Code' commands have long been in the X-10 suite, however they have not been implemented by manufacturers until recently.

The X-10 dimmers (LM14a and LM15a) as well as the new AM14a appliance modules are capable of transmitting as well as receiving commands.

The extended code commands are used for setting certain 'modes' within the modules, as well as for reporting their current 'status'.

With version 2.1.5 and later, XTension offers some verbs for processing and controlling devices that employ the extended code commands.

Please note that only the LynX-PLC and the CM11 can properly send and receive these extended codes. Therefore, in the logic of XTension, where the received extended code is reported, IF the controller is not a LynX-PLC, or CM11, then the 'current value' of the addressed unit will not be changed.

If the Controller is a LynX-PLC, or a CM11, then the command will be processed, and if it indicates a change in the status of the unit, the current value in the database record will be changed to reflect this.

The 'xpress' command is used in order to set the internal 'preset' level of these new 'smart' dimmers.

For example: If you set the internal preset of a LM14 to 50%, from then on, it will respond to a normal ON command by brightening gently to that level. And it will respond to a normal OFF command by dimming to zero.

The greatest advantage of this is that we avoid the necessity of sending multiple DIM/BRI commands. This reduces the number of commands on the powerlines, and the response of the module is somewhat faster.

The 'xpress' command can be used with the Leviton wall-mount dimmer switch (6381), to set its internal preset level. However it cannot transmit X-10 signals. Therefore, it cannot send its current status. This means that it will not react to the 'enable extended ack' commands.

### **xpress: Set the internal 'preset' dimming level**

**xpress** string -- unit name  
[**to** integer] -- percentage 0 - 100 of 'full ON'  
[**by** integer] -- [+ or -] percentage 0 - 100 of 'full ON'  
[**in** integer] -- seconds to delay before sending command  
[**for** integer] -- seconds to delay before reversing the same action

*example : xpress "Foyer Lights" to 25*

*forces the device (like a LM14a) to set its internal preset to 25%, and causes it*

*to softly change its brightness to that level.*

### **xtend: Send explicit extended code with data**

**xtend** string -- Unit Name  
[**data** integer] -- 8 bit data value from the X-10 Extended Code documents  
[**command** integer] -- 8 bit command from the X-10 Extended Code documents

*example : xtend "LM14" data 1 command 59*

*(Sends exactly the data byte = 1 and command byte = 59 to the address of the unit)*

### **enable extended ack for: Enable the auto-response for changes**

**enable extended ack for** string -- unit name

*example : enable extended ack for "Foyer Lights"*

**disable extended ack for: Disable the auto-response for changes**

**disable extended ack for** string -- unit name

*example : disable extended ack for "Foyer Lights"*

**ExData of: Retrieve last Extended code data from this unit**  
**ExCommand of: Retrieve last Extended code command from this unit**

*example : write log "Last Ex data was " & exdata of "LM14"*

*example : write log "Last Ex command was " & excommand of "LM14"*

Please note that the X-10 interface called the TW-523 which is employed by the LynX-10 and the "Two-Way" controllers WILL NOT receive these commands properly.

Further, early models of X-10 Repeaters become confused by these.

The CM11/Activehome interface will send and receive these codes, and they will be handled properly by that version of XTension.

=====  
It is a very interesting feature of the X-10 LM14 and AM14 that they will return some special status bits in the upper two bits of the extended data.

The Upper bit (7 = 128), is set to a ONE if the LOAD is connected.

IE: IF the light bulb is OK, or if there is some real resistive load.

IF the light bulb is burned out, or there is no resistive load,  
the bit will be zero.

So, if you wish to know whether there is a load connected, (or if the light bulb is burned out), just use the 'exdata' command :

```
if exdata of "LM14" > 127 then
  write log "The light bulb is OK"
else
  write log "The light bulb is burned out!"
end if
```

Bit 6 of the exdata is always a Zero if the module is a LM14, and always a One if the module is an AM14.

ONLY the lower 6 bits of the exdata represent the dimmer value.

=====  
**Please note** that there is a great deal of confusion about just which module or manufacturer uses the 'extended codes' or the 'preset dim' commands.

Some manufacturers employ the 'Preset Dim' commands to set the internal preset dim of their dimmers. Others use this command to handle such things as 'thermostats'.

X-10 corp chooses to employ the 'Extended Code' commands to set the internal preset level of their new dimmers (LM14/LM15), and to report their status.

Please do refer to the archives of the XTension Discussion List which are found on our website at [www.shed.com](http://www.shed.com). And don't forget that you can always subscribe to the list and get wonderful help from all of our users!

# What does XTension do when it receives an X-10 Extended Message ?

The "Extended message format" includes in its 'payload' :

Command	-- 4 bits
House Code	-- 4 bits
Unit Code	-- 4 bits
Extended command	-- 8 bits
Extended data	-- 8 bits

Whenever XTension receives an extended message, from a unit address that is in the your database :

- 1.The unit record fields "ExCommand" and "ExData" are conditioned with the raw 'command' and 'data' bytes from the message.
2. The variable 'future value' is set equal to the 'raw' 8 bits of the 'extended data' byte (same as above).
3. If the lower 6 bits of the 'extended data' are non-zero, then the Unit ON script is called if it exists. If the low 6 bits are all zero, then the Unit OFF script is called if it exists.
4. At entry to your Unit Script, you have the following variables :

**thisUnit** -- name of this database unit (Hall Light)  
**command** -- an integer -- the X-10 command that caused this trigger \*  
**wireless** -- T/F flag whether this command came in via the MR26  
**future value** -- the 'raw' extended message 'data'

**ExCommand** of "Hall Light" -- 8 bit extended message 'command' byte  
**ExData** of "Hall Light" -- 8 bit extended message 'data' byte. \*\*  
last **timestamp** of "Hall Light" -- last time this unit changed  
**value** of "Hall Light" -- the 'current value' of the unit,  
before this message occurred.

5. The user may change '**future value**' before exiting the script.
6. On return from the Unit script, XTension tests '**future value**'.  
If '**future value**' was not changed, and the 'extended **command**' was either a 49 or 56, then the message is assumed to be one of the known dimmer products that announce their current level via the extended code.

\*\*\*

In this case, the lower 6 bits of the extended data are 'scaled' from the 0-63 value, to 0-100, representing the current brightness percentage.  
And the unit record in the database is updated to this new 'current value' :  
$$\text{percent} = \text{integer value of } ((\text{raw} * 1.5873) + 0.5)$$

If the user script changed 'future value', then that value is stored in the unit record as 'current value'.

=====

CAVEATS:

\* there is a table of 'command' values in the XTension manual : Reserved Variables

\*\* this is redundant with 'future value'. But future value is transient, and ExData persists in the unit record until the next Extended message is received.

\*\*\* This will be done only if the X-10 controller is a LynX-PLC or CM11.  
Other controllers do announce partial extended messages, but they are  
only announced in the log. No processing will be done in response to  
them.

=====

Note also that I have not published a table of all possible extended command  
type codes. There is a document, and some manufacturers have published  
the code set that they use.





# Special Scripts and Named Values

**XTension** allows you to create 'special-named' scripts. Simply creating a "Global Script" with these names, causes **XTension** to execute them at their appointed times, or whenever certain system level events occur.

These Script Names are **RESERVED** for their respective intrinsic functions:

## Startup Script / Shutdown Script

On starting up, **XTension** will automatically execute the '**Startup Script**', and conversely, the '**Shutdown Script**' is executed anytime that you 'Quit' nicely out of **XTension**.

Obviously, you might want to always check things out or set up units into some known state on startup.

On shutdown, you might want to 'safe' some units or transfer control to another system.

These two scripts must be created as 'global' scripts. They must be named precisely as above.

## Sunrise/Sunset

These script names are reserved for suntime events. There is a chapter about them (Sunrise and Sunset...).

## Errors Script

This script is called whenever a significant error occurs. Before it is called, **XTension** sets a **global variable** which can be tested by the Errors Script : **Last Error**

## AUXRemote

This script is called whenever a command comes in via the MR26, that is encoded as a "MP3 remote" type of command.

This is particularly useful with the X-10 "MP3" wireless remote (UR51A) which sends over 50 unique 'button' commands.

Whenever **XTension** detects a "MP3" type command from the MR26, it will look for a Global script named "AUXRemote".

If you have created such a script, it will be called. Before calling the script, **XTension** sets the special variable 'command' to the 'button' number from the UR51A.

You can serve any of those buttons with commands, and ignore any that you don't need.

This is a great way to add up to 50 macros to your remote control system, without having to dedicate database units and X-10 addresses !

=====

## Values of Last Error :

**1 = The disk in use is full -- Logging has been disabled**

**2 = The Powerline interface failed to respond to a 'ping' (comatose?)**

**An attempt has been made already to regain comms with the controller.**

=====

### **Power Fail (LynX only)**

This script is called whenever it is known that a power fail has occurred. It is currently only available with the LynX versioni of XTension.

This script is called whenever the LynX sends a special message to XTension. This is most valuable when the LynX is put on the same UPS (battery backup) as the Mac running XTension, and the TW523 is plugged into the normal house 'mains'.

### **Additional special variables:**

#### **rude shutdown -- (not yet implemented)**

This variable is set at every startup. If the last shutdown of XTension was due to a powerfail or system crash, this is set to TRUE. If XTension was politely Quit, this is set to FALSE.

This can be used to determine whether you want to do anything special due to a 'rude shutdown'.

example : if (rude shutdown) then  
          write log "The last shutdown was RUDE !"

#### **startup time -- (not yet implemented)**

This is a timestamp of the last time that XTension started up. It is for information only, and simply gives some knowledge of how long XTension has run without failure or restart.

example : write log "Been up since " & (startup time)

#### **last error -- (not yet implemented)**

This is the number of the last 'protocol' error that XTension encountered while dealing with the X-10 interface. It is useful only for custom diagnostics which keep track of this kind of error.

#### **error time --(not yet implemented)**

This is the timestamp of the 'last error' above.

example: write log "Last interface error was at (error time)

#### **port status --(not yet implemented)**

This is a true/false indicator of whether the serial port is enabled.

It can be used in custom scripts which manage multiple copies of XTension and whether to enable or disable the serial port.

( see the 'set port' verb for enabling/disabling the port)

example : if port status is true then  
          write log "The serial port is enabled"  
          end if

## Sunrise / Sunset Scripts and Times

**On starting up, and every day at midnight**, XTension will automatically calculate the sunrise and sunset times for your location !

( There are two verbs which allow you to offset both the sunrise and the sunset times by +/- 90 minutes ! see 'adjust sunrise/sunset' )

If you have properly set your latitude and longitude for your location using the [Map control panel of your Mac](#), and your Mac's [clock](#) is correct, then XTension will do a reasonable job of calculating the expected times of sunrise and sunset.

**Every day** at the calculated sunrise time, XTension will execute a global script named '[sunrise](#)' if you have created one. Likewise at the expected sunset time, a script named '[sunset](#)' is executed.

If there are things that **you** always want to do at sunrise and/or sunset, then you must create these scripts and include the instructions you want.

### Sunrise/Sunset times and whether it is 'daylight'

XTension maintains a set of internal variables which hold today's '[sunrise](#)', and '[sunset](#)' times.

These variables represent the number of [seconds after midnight](#) of the current day when sunrise and sunset will occur at the latitude/longitude that you have entered in your Map control panel.

In addition, XTension maintains a variable named '[daylight](#)', which can be used in conditional scripts.

You **cannot** set or modify '[daylight](#)' directly, this is done automatically by XTension. However, you can adjust the **suntimes** by the 'adjust' verbs below.

#### **Examples:**

```
If daylight is false then
    turn on "Front door light"
end if
```

----

```
write log "Sunrise occurs today at " & time string of(sunrise)
```

-----

```
write log "Sunset will be at " & time string of(sunset)
```

(PS: yes, you do need the parentheses around the words...)

### Calculating sunrise or sunset for a given Date :

Version 3.0.9 added two verbs which can be used to calculate the sunrise and sunset times for any given date.

#### [sunrise for:](#) Returns the Time of sunrise for the given date

**sunrise for** date -- any valid specification of a DATE in Macintosh format.

returns : Date/Time of sunrise

#### [sunset for:](#) Returns the Time of sunrise for the given date

**sunrise for** date -- any valid specification of a DATE in Macintosh format.

returns : Date/Time of sunset

example :

```
set theDate to date "12/25/2000"
```

```
write log "Sunrise for Christmas will be at : & sunrise for theDate"
```

## Special Processing Scripts

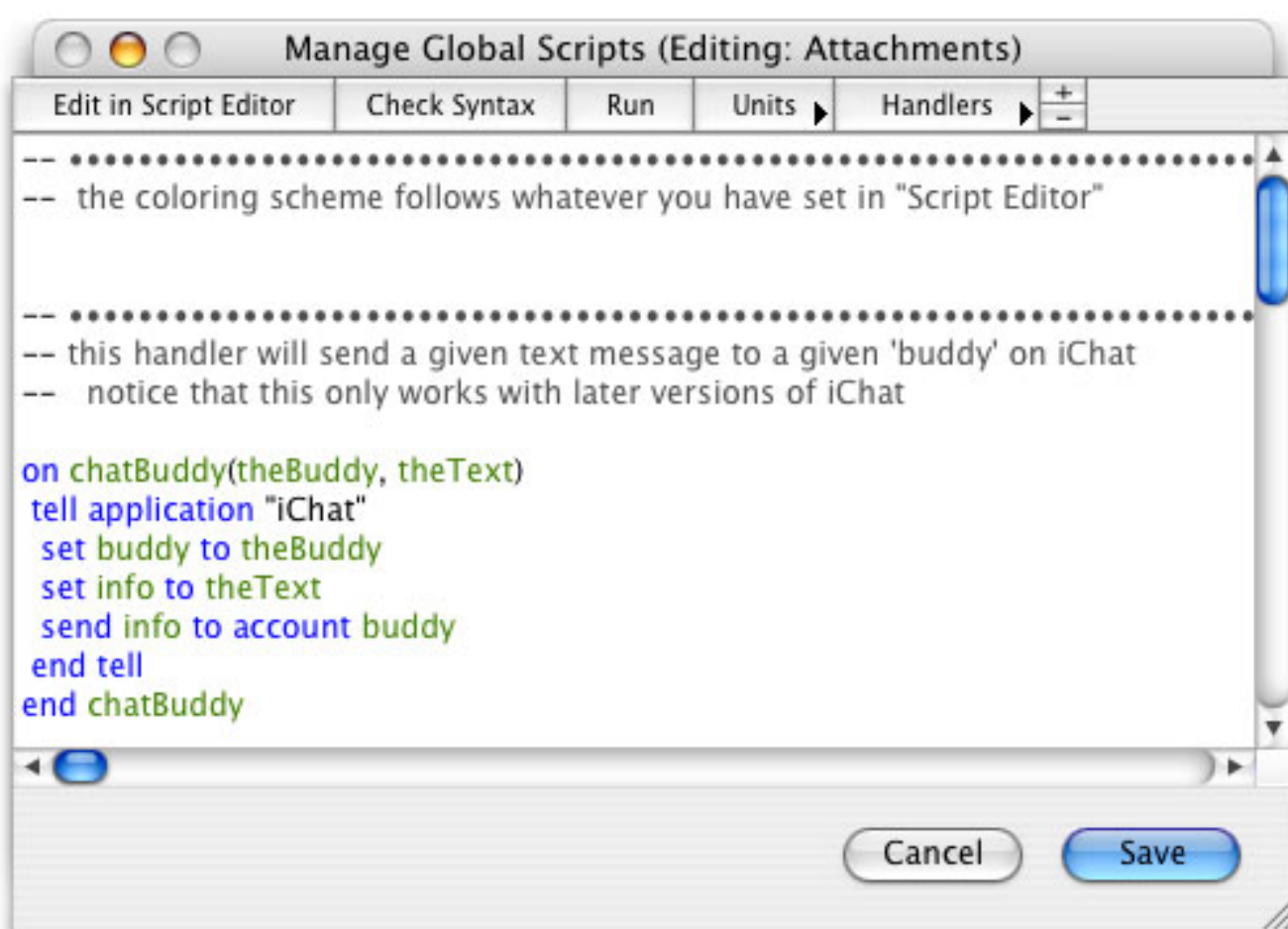
### Attachments -- must be in the same folder/level as XTension:

This is the '**hook**' in XTension which allows you to create your own verb 'handlers', or alter the 'handling' of existing XTension verbs.

If you are adept with AppleScript, you can create functions which integrate other applications or peripherals, and thus your own custom interface to your home automation system.

Please see the articles and technotes on the shed.com website which offer help and insight into other uses for this special script.

Following is a simple example of just how you might create a pair of 'handlers' which would allow you to set and query a value of any type.



Note that when XTension saves the database, it always saves the current 'context' of the Attachments script. This is one way to create lists, values, text strings, reports etc which are 'outside' the current scope of XTension's capabilities...

## Interface Service Verbs

### Enable or Disable the Serial Ports:

#### **available ports** : *Get a list of available 'ports'*

**available ports**    *a list (read only)*

This verb returns a list of strings with the names of serial ports or other places that you might connect an interface.

#### **interface type** : *Set or Get the local X10 interface type*

**set interface type**    *string [ CM11 or LynX]*

**get interface type**    returns **string** [ CM11 or LynX]

This verb is used to set or get the powerline interface type. Performs the same function as setting same in the Communications Preferences dialog.

#### **wireless interface type** : *Set or Get the local wireless interface type*

**set wireless interface type**    *string [ MR26 or W800]*

**get wireless interface type**    returns **string** [ MR26 or W800]

This verb is used to set or get the wireless interface type. Performs the same function as setting same in the Communications Preferences dialog.

#### **local X10 port** : *Get or Set the port for the powerline interface*

**set local X10 port**    *string*

**get local X10 port**    returns *string*

This verb lets you get or set the string name of the port assigned to the powerline interface.

Returns a string name of a device port, as was selected in the comms prefs, or 'set'.

NOTICE that any changes do not take place until after the next time the port is 'enabled'.

#### **local wireless port** : *Get or Set the port for the wireless interface*

**set local wireless port**    *string*

**get local wireless port**    returns *string*

This verb lets you get or set the string name of the port assigned to the wireless interface.

Returns a string name of a device port, as was selected in the comms prefs, or 'set'.

NOTICE that any changes do not take place until after the next time the port is 'enabled'.

#### **set port** : *true/false*

**set port**    *boolean true or false*

This verb is used to enable or disable the X10 powerline interface service. This will cause the load or unload of the Interface Service Function for the selected X10 controller.

#### **set rf port** : *true/false*

**set rf port**    *boolean true or false*

This verb does the same for the MR26 or W800 Interface Service Function.



## **Send serial data:**

### ***send data: Send raw data out the (X10) port***

**send data** string -- *Data to send*

*example : send data "+++ATDT911"*

This function is for troubleshooting and special X10 interface commands which are not supported by XTension. The data to send is unrestricted, but obviously must make sense to some device which is connected to the chosen port.



# Special features for the LynX-Port

## What is the LynX-Port ?

The LynX-Port is the big-brother to the LynX-10.

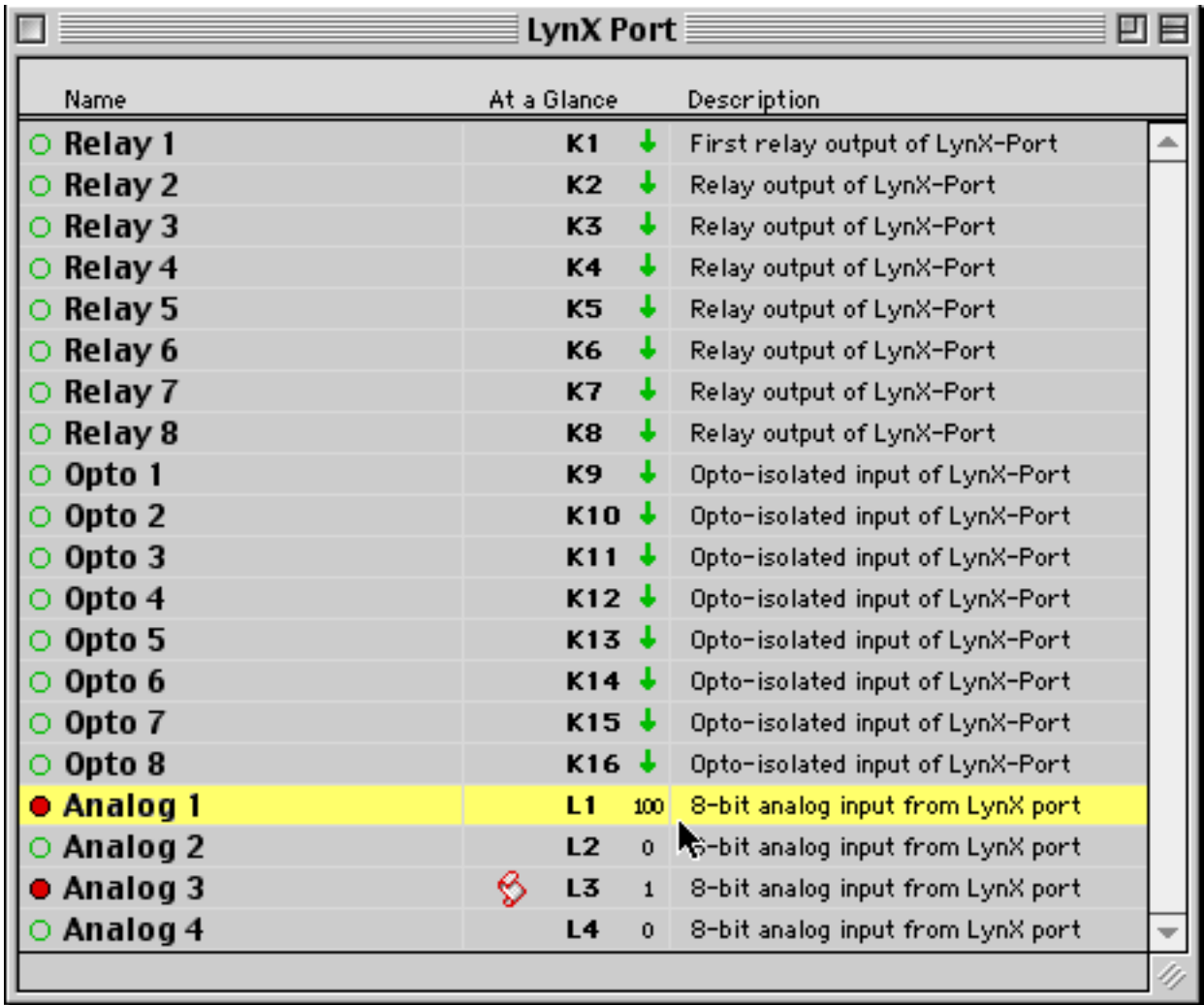
The LynX-10 is the most reliable X-10 controller that is supported by XTension. Its reputation is unparalleled.

The LynX-Port combines the reliability of 'hard-wired' elements with the ease and low cost of the X-10 system.

In addition to the normal X-10 features of the LynX-10, the LynX-Port offers :

- 8 Opto-isolated contact or voltage sensors.
- 8 DPDT relay (contact) outputs.
- 4 Analog inputs with 8-bit resolution (0-255)

Here's an example List window showing all of the units that are part of the LynX-Port if the Base house code has been set to "K" :



Name	At a Glance	Description
○ Relay 1	K1 ↓	First relay output of LynX-Port
○ Relay 2	K2 ↓	Relay output of LynX-Port
○ Relay 3	K3 ↓	Relay output of LynX-Port
○ Relay 4	K4 ↓	Relay output of LynX-Port
○ Relay 5	K5 ↓	Relay output of LynX-Port
○ Relay 6	K6 ↓	Relay output of LynX-Port
○ Relay 7	K7 ↓	Relay output of LynX-Port
○ Relay 8	K8 ↓	Relay output of LynX-Port
○ Opto 1	K9 ↓	Opto-isolated input of LynX-Port
○ Opto 2	K10 ↓	Opto-isolated input of LynX-Port
○ Opto 3	K11 ↓	Opto-isolated input of LynX-Port
○ Opto 4	K12 ↓	Opto-isolated input of LynX-Port
○ Opto 5	K13 ↓	Opto-isolated input of LynX-Port
○ Opto 6	K14 ↓	Opto-isolated input of LynX-Port
○ Opto 7	K15 ↓	Opto-isolated input of LynX-Port
○ Opto 8	K16 ↓	Opto-isolated input of LynX-Port
● Analog 1	L1 100	8-bit analog input from LynX port
○ Analog 2	L2 0	8-bit analog input from LynX port
● Analog 3	L3 1	8-bit analog input from LynX port
○ Analog 4	L4 0	8-bit analog input from LynX port

## How does XTension handle the LynX-Port ?

XTension attempts to integrate the 20 different inputs and outputs just like they were 'normal' Database Units.

You turn on/off the discrete outputs, and you expect to respond to incoming commands from the 'inputs'.

Just the same as if the discrete outputs were "Universal Modules", or simple Appliance modules, and the inputs are like "PowerFlash Modules", motion sensors, or manual remote controls.

You create database entries with the usual 'names', and you assign their

addresses according to the '**Base House Code**' that you assign to the LynX-Port.

You do this either with the special LynX script verb :

**configure base address to "C" [A-P]**

**For example:** If you configure the LynX-Port base house code to house code "C ", then the special ports would be configured to these addresses:

Outputs 1 - 8 >> C1 thru C8  
Inputs 1 - 8 >> C9 thru C16  
Analog 1 - 4 >> D1 thru D4

Thus you should reserve all of that address range, even if you do not expect to use all of the ports.

NOTE that whatever house code you choose for the 'discretes', the four analog ports will occupy the next-higher House code. IF the house code is chosen to be "P", the Analogs will take "A" ...

You might connect a simple micro-switch to a doorway, and run the two wires to the screw-post inputs of Input #1 of the LynX-Port.

Create a New Unit in XTension.  
Name it "Door Switch"

Create unit scripts for the ON and OFF states, just like you would for a motion sensor, testing time of day or house 'mode', and turning on lights or alarms accordingly.

Likewise create New units for each of the contact outputs that you need, and simply turn them On or Off with other scripts or scheduled events.

turn on "Garage Door Opener" for 1

Note that the LynX-Port does offer options for turning off any output relay automatically after a selected time.

Further, there are many options that combine the operation of different relays with the opto-isolated inputs.

### **What really happens with the inputs and outputs ?**

In the case of Output relays, You send a ON command to the LynX using the standard X-10 address scheme.

turn on "Garage Door Opener"

The LynX-Port closes the output relay, and also sends a X-10 command to the powerlines:

House C Unit 1 ON

Likewise, when one of the Input ports changes state, the LynX-Port sends **both** a message to XTension, and a corresponding X-10 command to the powerlines.

### **How do we handle the analog inputs ?**

XTension provides a verb that forces the LynX-Port to sample each of the 4 analog channels.

[sample all analogs](#)

This command simply tells the LynX-Port to start the sampling process. The LynX-Port responds with 4 separate messages that announce the current 'raw'

readings of the analog ports.

NOTE: These messages do not come in immediately. The script that includes the 'sample all analogs' command should not expect the results to appear within the scope of that script.

Whenever the LynX-Port sends the messages, XTension sees these and 'triggers' the appropriate Unit Scripts, IF you have created any.

Like other 'variable' units, (not on/off), XTension triggers the unit ON script if the new value is greater than Zero. It triggers the unit OFF script if the new value is Zero.

Please note that when the Unit script is called, the value received from the analog port, is contained in the variable "Future Value".

You have the option of 'scaling' that value as you wish and putting any value into "Future value". When your script finishes, whatever is in Future Value, will be put into the 'current value' of the unit in the database.

XTension also changes the current value of the corresponding unit in the database according to the 'raw' value that was received.

Unit scripts are expected to 'process' the data into reasonable units of measure, like 'temperature', voltage, humidity etc.

There are two tutorials about Analog data processing on the shed.com website. Please refer to these as well as the documentation you get with the 'sensors' that you purchase.

see: <http://www.shed.com/articles/TN.ADBIO.FEP.html>  
and: <http://www.shed.com/articles/TN.analog.html>

### **Special verbs for dealing with the LynX-Port**

XTension provides verbs for configuring the basic elements of the LynX-Port. Many other options are available using the "Send Data" command, and no doubt, many more verbs will be added...

### ***configure base address to : Sets the base house code of the LynX-Port***

**configure base address to** string -- Valid X-10 house code [ A - P ]  
*returns : nothing*

*example :*  
*configure base address to "C"*

### ***sample all analogs: Force the LynX-Port to sample the 4 analog inputs***

**sample all analogs** -- No parameter needed  
*returns : nothing*

*Causes 4 messages to be sent to XTension which will trigger any of the ON/OFF unit scripts for the 4 different Analog units.*

## Special features for the LynX-PLC

### What is the LynX-PLC ?

The LynX-PLC is the latest (June 2001) version of the classic LynX co-processor series from Marrick Ltd. ([www.marrickltd.com](http://www.marrickltd.com))

This version includes many new features, including the ability to send the 'extended codes' which are necessary for controlling the 'smart' dimmers which have internal preset dim features.

(like the X-10 "LM14a" and Leviton 16383)

The most significant thing about the LynX-PLC is that it no longer requires the TW-523 'helper' interface. The 'PLC' looks very much like the X-10 CM11.

In addition to supporting the 'legacy' LynX protocol, the PLC also offers a new "LynX-Net" protocol which promises to support different 'future' home automation devices.

XTension supports only the 'legacy' protocol at this time, but as soon as these new devices appear on the market, XTension will be changed to support the new protocol.

XTension will automatically recognize the PLC on startup, and will configure it properly.

### XTension features supporting the LynX-PLC :

### Setting the Transmitter and Receiver controls:

**These controls are really for those who want to play with the intimate controls of the LynX-PLC. Please do not change these items unless you know what you are doing.**

The transmitter output register controls the amplitude of the powerline signals that are transmitted from the LynX-PLC.

Likewise, the Receiver sensitivity option controls the 'hearing' of the PLC.

These are great options, but they also imply some intelligence in the choice of these values.

The most important consideration is that it is possible to set these levels so that the transmitter overdrives the receiver directly, so any command to send to the powerline, causes the receiver to generate gobs of errors.

But properly controlled, these new options can be of great value to home automation enthusiasts.

Consider being able to, for a moment, turning off the X-10 receiver, and turning the transmitter up to some 'known good' level, and then transmitting one X-10 OFF command to that troublesome lamp at the Gazebo that is 500 feet from anything.

There is a manual method of piddling with your home system to discover just the right 'tweaking' for your normal environment, as well as finding the right settings for those troublesome remote devices.

Consider setting the receiver level to zero. Then trying to command that Gazebo.



Continue to increase the Transmitter output level and re-trying until you find a 'reliable' level that will command that Unit.

Thus, whenever you need to be more certain that that Unit had a better chance of seeing the command, you just need to first shut down the X-10 receiver, jack up the X-10 transmitter, and then issue the command.... and then reset the transmitter and the receiver to their normal settings.... whew !

Now that's a possibility with the current verbs, and future versions of XTension may provide more automatic or perhaps intuitive handling.

XTension offers two verbs that make this easy to do in your scripts. PLEASE understand that this is a very dangerous tool set. You can set values with these verbs that will totally screw up your home system.

Not physically, but you can put it into a state where it is very difficult to understand just why nothing is happening correctly :-)

**set PLC output to : Set the X-10 transmitter output level**

**set PLC transmitter to integer** -- [0 - 100] percentage of output maximum

*returns : nothing*

*example :*

*set PLC transmitter to 75 -- sets the LynX-PLC transmitter to 75%*

**set PLC receiver to : Set the X-10 receiver sensitivity level**

**set PLC receiver to integer** -- [0 - 100] percentage of maximum sensitivity

*returns : nothing*

*example :*

*set PLC receiver to 75 -- sets the LynX-PLC transmitter to 75%*

**initialize controller : Set transmitter and receiver settings to factory defaults**

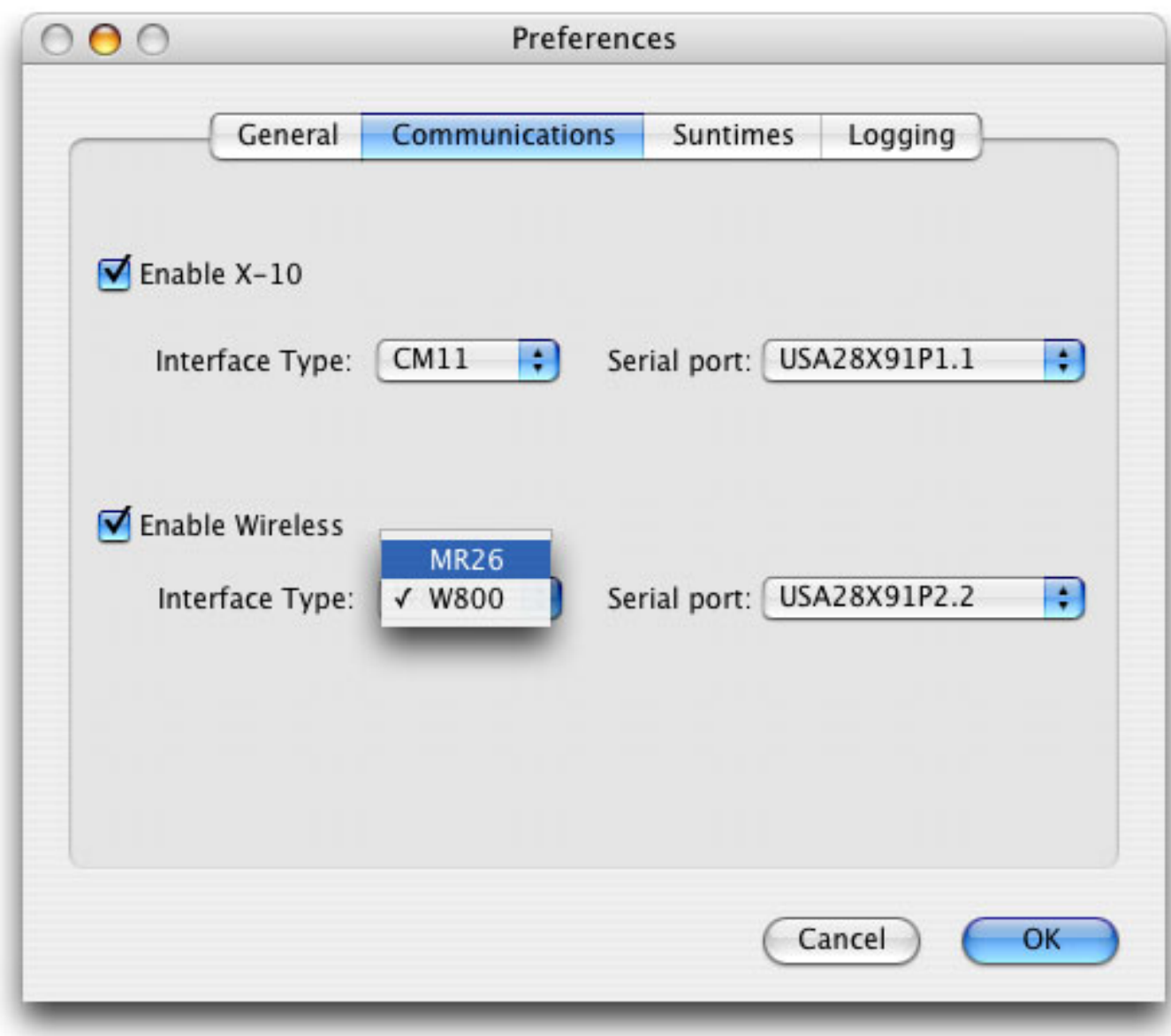
**initialize controller**

*returns : nothing*

This attempts to set the Receiver and Transmitter to 50%

# Special features for the MR26 and W800

## Choose the MR26 or W800 from the Communications Preferences panel



## Why do you care ?

Traditional X10 powerline systems sent RF signals to a Powerline Transceiver which then re-transmitted those signals onto the powerlines. This caused additional loading and collisions on the lines.

About spring of 2001, X10 came out with a RF receiver that passed wireless signals directly to a serial RS232 channel as opposed to sending them to the powerlines.

Since these signals require much less time to transmit, and do not appear on the powerlines at all, this represents a great change in the technology, making it far more responsive, and much more reliable.

Sometime in 2003, WGL Designs in Texas produced a third-party product that replicated what the MR26 does, and adds the ability to transceive the signals from the X10 Security System modules which the MR26 does not do. (see [www.wgl designs.com](http://www.wgl designs.com))

In addition to passing signals directly to a serial channel, the MR26 and W800 transceive signals from all X10 house codes as opposed to the old-style single-house code transceivers like the RR501 and TM751.

The MR26 will receive all commands from the X-10 "MP3" remote (UR51A). Almost every key on that remote will generate a special command that XTension will recognize. XTension will then attempt to call a special named script "AUXRemote". With the 'button number', the script can then determine what special thing is desired by the

press of this button... like maybe triggering some ZephIR commands ?

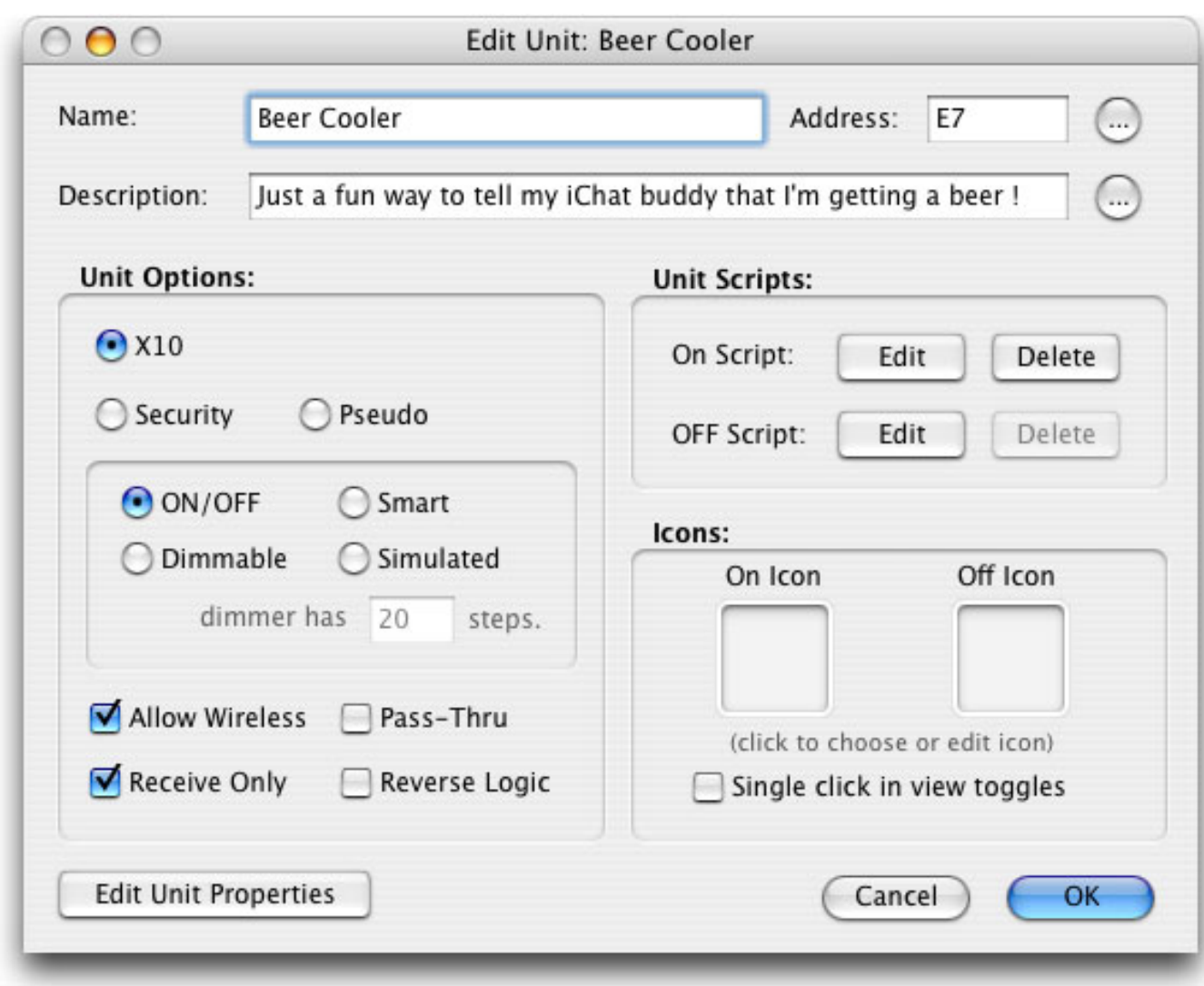
The MR26 can be purchased for about \$30US from X-10.com, the W800RF32 costs about \$80 with a very good antenna.

Be sure to visit the website tutorial:  
[www.shed.com/tutor/mr26.html](http://www.shed.com/tutor/mr26.html)

**Configuring a Wireless Unit**

If the unit is a normal X10 wireless device, just choose the **X10** button, and check the **Allow Wireless** option...

If the unit is a X10 security device, click the **Security** button.  
( Also see the chapter on Wireless Security Units. )



XTension will thereafter allow signals received from the MR26 or W800 to be recognized.

You **MUST** do this for every unit from which you expect to receive RF signals. If a message from the wireless controller is for an address that is not opted to be 'wireless', it will be logged as ignored.

This does not preclude you from having wireless units that are handled by standard X-10 RF transceivers like the TM751 or RR501.

Note that when a unit is configured to be "Wireless", it will show the 'Antenna' icon in all list views...

**You can also enable/disable the MR26/W800 from a script:**

### **set rf port : true/false**

**set rf port**    *boolean    true or false*

### **You can also modify the 'time filter'**

### **change MR26 time filter to xx :integer number of 'ticks'**

**change mr26 time filter to [integer]**    *value between 1 and 120*

(This works for the W800 also)

This should only be necessary when you are running XTension with a MR26, on a Mac that is also running other processor-hungry applications. In such case, it could be that you will see more than one or two instances of the same command from a wireless device. These are normally sent by the device, but XTension tries to 'filter' them so that only one is actually processed.

If you are seeing an unusual number of these, you might try to increase the 'time filter' from the default 40 'ticks'.

Note that a 'tick' is 1/60th of a second. Wireless devices typically send 3 or more of the same command for the same stimulus.

These are sent in rapid succession, but it could be that other apps delay XTension processing. Increasing the number of ticks will reduce the probability of these repeated commands.

However, it will also reduce the responsiveness of the technique of holding down a DIM/BRI key on a manual remote.

### **Reduce the number of 'phantom' signals**

If you find that you are receiving an unusual number of false signals from the wireless controller, you can 'suppress' them at the risk of losing faint but real signals. This does not affect signal sensitivity in the controller, but rather requires that multiple contiguous signals be received before they are counted as valid. This can be a very handy thing in some electrically noisy homes.

### **suppress wireless phantoms**

### **allow wireless phantoms**

The 'allow' verb simply negates the suppression method.

### **NOTICE about XTension processing of wireless commands:**

XTension does not by default pass-thru wireless commands to the powerlines.

This is specifically different from standard wireless commands that are transceived by the TM751 and RR501 X-10 transceivers. In the standard case, by the time that XTension sees the command, it has already been put on the powerlines and will have been seen and responded to by any X-10 module with the same-address .

**If you wish to have wireless commands automatically passed-thru to the powerlines, you must select the 'Pass-Thru' option in the Edit Unit dialog of the respective unit.**

**You will really want to see the tutorial about the MR26 which is on the shed.com website: <http://www.shed.com/tutor/mr26.html>**

ALSO see the chapter about special Command Variables... OR HOW you can

determine that the stimulus to a unit script was due to a 'wireless' command ...  
!





## Special script variables

### What are 'reserved name variables' ?

XTension provides certain variables that are available within your scripts, that indicate various processing status etc.

These variables are referenced by name, without quotation marks.  
See examples below.

### command, wireless, future value, thisUnit, thisScript etc.

Before running any Unit Script, XTension will set 4 important variables that you can use in conditional tests.

NOTE: These variables are valid at the beginning of any Unit Script, up until the time that you stimulate any other database Unit within that script. As soon as you command any other unit, these variables are set for the commanded unit, thus the variables are valid for that unit and do not have relevance to the original Unit.

Thus if you want to use these variables 'later' in your Unit Script, you should save them into other script variables at the beginning of the Unit script...

### The 'thisScript' variable

This variable is set by XTension to the NAME of the Global Script, whenever a Global Script is executed. It is set to NULL immediately after the script is finished. Like the 'thisUnit' variable, it is probably best to enclose it in parentheses.

It is intended only to make it easier to write scripts that refer to themselves.

Example:

execute (**thisScript**) in 5 \* minutes

### The 'thisUnit' variable

This variable is set to the NAME of the Unit that owns this script. The most common use for this is to be able to more easily copy/paste scripts between different Units.

Because of the possibility of mis-interpretation of this string, you may have to enclose it in parentheses.

Example :

turn off (**thisUnit**) in 5 \* minutes

### The 'command' variable

This variable is set to an integer value that equates to the actual X-10 'command' that caused the Unit Script to be executed.

The most common commands are :

**ON** = 3

**OFF** = 4

**DIM** = 5

**BRI** = 6

The complete command code list is below... \*\*

Thus, in a Unit Script for "Lamp", you could say :

```
if the command = 3 then
    write log "It was an ON command"
end if
```

### The 'wireless' variable

IF you have installed and enabled the MR26, you might want to be able to distinguish between commands that came from the MR26/W800 (wireless), and those that came in from the powerline (standard X-10).

If the command came in from the MR26, XTension will set the 'wireless' variable to TRUE. Else it will be set to FALSE.

IE:

```
if wireless then
    write log "It was a wireless command"
else
    write log "It was a powerline command"
end if
```

### Putting these together:

In the ON script for a "Lamp", that you command from wireless remote or sensor, you might want to do this:

```
=====
if wireless then
    if (the command) = 3 then turnon "Lamp" with no script
    if (the command) = 4 then turnoff "Lamp" with no script
    if (the command) = 5 then dim "Lamp" by -5 with no script
    if (the command) = 6 then brighten "Lamp" by 5 with no script
end if
```

```
-- Note that the article 'the' is optional in AppleScript.
-- You could just say: if command = 3 ...
```

```
=====
```

Remember, that with early implementations of the MR26, XTension does NOT automatically pass commands from the MR26 to the powerlines. You must decide whether you want to do this.

### The 'future value' variable

This variable is set by XTension to the value that the Unit will be set to AFTER the Unit Script has completed.

Like the wireless and command variables, this one is temporary, and valid only until another Unit has been commanded by the Unit Script.

This feature gives you the ability to compare the 'previous' value, with the 'new' value.

IE:

```
if future value > (value of "Lamp" + 50) then
    write log "Gosh, that's a lot !"
end if
```

When XTension calls the Unit script, the Future Value is set appropriately. But DURING this unit script, and only for the Preset Dim and Extended Code, commands, you may use the verb :

**change future value to [integer]**

**For example, you might want to 'scale' some raw value to some 'processed' value, such as Temperatures etc.**

In these cases, IF 'future value' is changed, the NEW value of the Unit will be set to the NEW value of Future Value...

=====

**\*\*The 'Command' variable : Possible values**

- 1 = All Units Off
- 2 = All Lights On
- 3 = OnCommand
- 4 = Off Command
- 5 = DimCommand
- 6 = Bright Command
- 7 = All Lights Off Command
- 8 = Extended Code Command
- 9 = Hail Request Command
- 10 = Hail Ack Command
- 11 = Preset Dim Command
- 12 = Preset Dim Address
- 13 = Extended Data Command
- 14 = Status On Command
- 15 = Status Off Command
- 16 = Status Request Command
  
- 33 = Toggle Command
- 34 = Execute Script Command
- 36 = Force On Command
- 37 = Force Off Command
- 46 = XPress Command
- 47 = Xack Mode Command
- 49 = Received Value For
- 51 = MP3 Command

**The 'Last Error' variable**

This variable is set by XTension to a value which represents a specific 'system' type of error, before calling the **"Errors Script"**...

=====

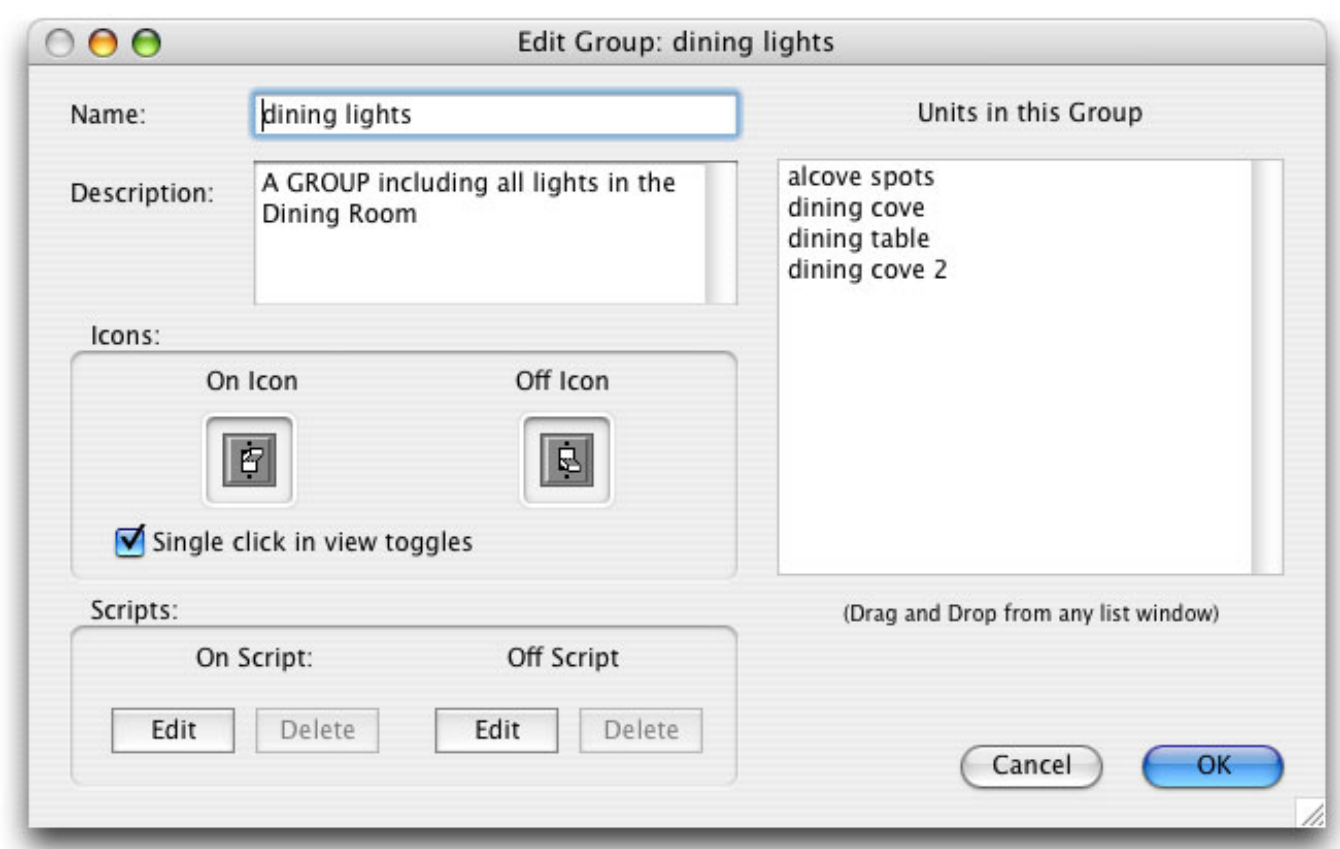
**Values of Last Error :**

- 1 = The disk in use is full -- Logging has been disabled**
- 2 = The Powerline interface failed to respond to a 'ping' (comatose?)**

**An attempt has been made already to regain comms with the controller.**

# Groups help reduce script steps

By creating **groups** of units, you can control several devices with a single XTension script or button.



Any command to the **group name** will cause XTension to issue the same command to **each** of the members of the group individually. Thus you can turn them all ON or OFF with a single :

“**turnon** “All Outside Lights”

Groups are also useful when you want to create a List of items of special interest, such as "All Motion Sensors".

For example, when you are modifying the scripts of a group of related units, you may want to create a limited List window in order to reduce the amount of scrolling you might have to do from the Master List window.

**The sequential order of the units in any Group is important:**

With versions of XTension after 1.7, it is possible to 'drag around' the units which are in a Group, thus 'ordering' them such that **you** can specify the **order** in which they will be 'commanded' whenever you issue a command to the entire 'group'.

## **Blocking helps reduce complexity**

You may ‘block’ or ‘unblock’ any unit or group. Blocking means that XTension will ignore any X-10 messages which come in addressed to that unit and will not allow any script to change or issue commands to that unit.

**block unit** "All Outside Alarms"

Unblocking reverses this logic, and allows normal behavior.

**unblock unit** "All Outside Alarms"

You may at any time issue the command ‘Unblock All Units’. This will re-enable any and all devices which may have been blocked.

You might wish to prevent any outside light from coming on while it was clearly ‘daylight’.

```
if Daylight is true then
  block unit "All Outside Lights"
end if
( this might also be done in the "Sunrise" script... )
```

If you had several chimes as alarms, you might want to block some of them according to your current location. If you're in the bedroom, then you may not want alarms going off all over the house.

```
( on movement in bedroom ) :
block unit "All Alarms"
unblock unit "Bedroom Chime"
...
```

Note that with XTension 3.1.0 and later, you can delay the action and reverse it with one verb :

```
( on movement in bedroom ) :
block unit "All Alarms" in 5 for 60
unblock unit "Bedroom Chime" in 60 for 5
...
```



## **Saving the Database**

Aside from the menu method of saving and archiving the database, there are also two verbs which can be used to do this within any script.

**save database:** Write the current database to disk immediately  
save database

**archive database at:** Save a copy of the XTension database at the specified File Spec address. (not yet available in OS X version)

**archive database at** file specification -- *a common file specification for the saved file*

**ex:** archive database at file "HD:Desktop Folder:XTensionf:Arch35"



## **Technical Notes**

( see also separate Technotes at [www.shed.com](http://www.shed.com) )

### **A note about memory usage:**

Graphic Views take up the most space, possibly more than all of the application and the rest of the database combined. You may use as many pictures as you like, but you will have to increase the run-time memory size of XTension.

### **A note about keeping the Macintosh running :**

You must keep both the Macintosh powered on at all times. If you really want to create a reliable security system, then you should invest in a battery backup system for your Macintosh.

The reasoning is that you want this system to stay running and alert at all times. XTension is a system which expects to run continuously. The features it provides need to be continuously aware of the state of all devices.

Should a power-fail occur, the Macintosh, XTension and the interface will recover properly regardless of whether you have a battery backup.

A startup script can be used to ensure that any devices which need to be reset after a power fail can be serviced at each startup.

Scheduled events are automatically recovered, and those which may have been missed due to the outage are announced.

( If they were missed by more than 10 minutes, they will not be performed. )

### **Bandwidth**

Because X-10 signals require about 6/10ths of a second for a single command, you must consider the number of commands that you initiate. When you create scripts and sequences, think about the number of commands that have to be sent to satisfy the intention of the scripts. Don't create a bigger problem. Reduce unnecessary X-10 bus activity. The best way to do this is to employ one of the wireless controllers and get all of that motion sensor traffic off of the powerlines.

Note that some X-10 interfaces differ in the the time required to process commands from the Mac. The version of XTension you have is adapted to the timing of the your interface.

### **Serial ports :**

In case you are interested in making your own cables, here are the maps :

#### **CM11 or CM10A :**

Pinout between the DIN-8 and the RJ22 :

DIN-8 to RJ22 (4)

- |   |   |                                             |
|---|---|---------------------------------------------|
| 5 | 1 | RxD to Mac from CM11                        |
| X | 2 | unused pin, any unused wire to fill plug(2) |
| 3 | 3 | TXD from Mac to CM11                        |
| 4 | 4 | Gnd                                         |

NOTE: the RJ22 is the very small 4 pin plug often found in telephone handset cords. The pin numbers above are relative to the position of the wires in the RJ22 :

Hold the plug with the tab down, and pointing away from you.  
Pin number 1 is leftmost, sequential from there.

**Lynx-PLC, LynX-10 or LynX-Port :**

Pinout between the DIN-8 and the DB-9 :

DIN-8	to	DB-9
1	7	RTS from Mac to LynX-10
2	8	CTS from LynX-10 to Mac
3	3	TXD from Mac to LynX-10
4	5	Gnd
5	2	RXD from LynX-10 to Mac

**"Two-Way" from Home Intelligence :**

Pinout between the DIN-8 and the DB-25 :

DIN-8	to	DB-25
1	4	RTS from Mac to TW
2	5	CTS from TW to Mac
3	2	TXD from Mac to TW
4	1 & 7 ?	Gnd
5	3	RXD from TW to Mac

It may be possible that a common Mac to DB25 Modem cable will work as is...?  
Note that the TW takes its power from the Mac handshake line RTS (pin 4).  
Thus you don't want to make a cable which is unusually long.

**CP290 :**

Pinout between the DIN-8 and the DIN-5 :

Note that this is a 'mini' DIN-8, and a standard DIN-5....

DIN-8	to	DIN-5
3	2	TXD from Mac to CP290
4	3	Ground
5	4	RXD from CP290 to Mac

...

## Trouble ?

---

If you have a CM11E :

If you can't get it to communicate on first attempt, or if it WAS working and now it doesn't : see the next chapter...

Is the serial connection solid?

Are the XTension communications settings right? (see Preferences)

Are two or more things conflicting with each other?

Are you trying to use the serial port which AppleTalk is using ?

Are you missing X-10 bus events?

Can you send a X-10 command from an X-10 wireless or wall switch? Does it do what it's supposed to but you don't see the event on the XTension Log Window ?

If not, then the signal strength of the X-10 module may be weak or electrical noise may be affecting the signals.

Are commands getting through?

Can you turn on/off a light?

If so, it says a lot about the setup of XTension and the controller and the cable...

If not, try putting the lamp or the interface into a different wall socket and try again. If it works some places but not others, then you may need a signal coupler/amplifier as seen in the home automation catalogs.

Things used to work but don't now...

If you have had a Mac system failure or something like it:

Take the XTension Preferences out of your System Folder.

(This was an old remedy, it may no longer be valid with 1.7)

IF nothing has happened with the Mac, then it is likely that something has happened to your home wiring or something new has been plugged in which is now blocking your powerline signals.

## Where to get help

We will be glad to help you understand how to use XTension.

## Trouble with your CM11A?

---

ONLY if you have a CM11A :

If you can't get the CM11A to communicate with XTension.  
on first attempt,  
or if it WAS working and now it doesn't :

Perform the following sequence exactly :

1. Unplug the CM11A from the wall socket
2. Unplug the serial cable from the Mac.
3. Make sure batteries are NOT installed
4. Leave the CM11A unplugged for ONE HOUR  
then:
5. Plug CM11A back in to wall socket
6. Plug serial cable back into Mac

To verify operation, tell XTension to send a command  
to any X-10 address.

Error messages will appear in the log if there is still a problem.

XTension does not have to be shutdown for this.

This may seem like Voodoo, but like other forms of magic,  
the steps must be followed carefully. And, like Voodoo,  
this will revive all but the most 'fried' unit, just as 'magically'  
as it quit working...

- There is a verb to '**Initialize Controller**' which will clear CM11 memory.

Sometimes the CM11 imagines that it has something stored in its  
EEPROM, like a scheduled event or 'macro'. Choosing this verb  
to clear the CM11 memory of these unwanted items.